COMP 3550

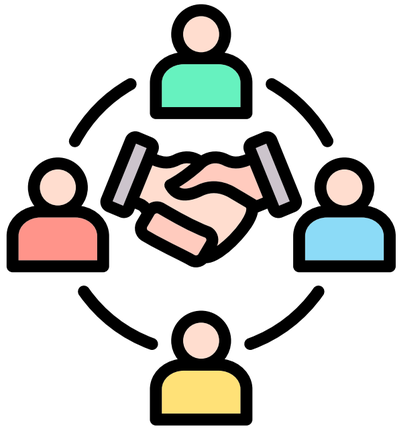
3.3 — INTRODUCTION TO TESTING & THE TESTING PYRAMID

Week 3: Version Control & Testing Foundations

WHY TEST?

Catch Bugs Early

Prevent Regressions

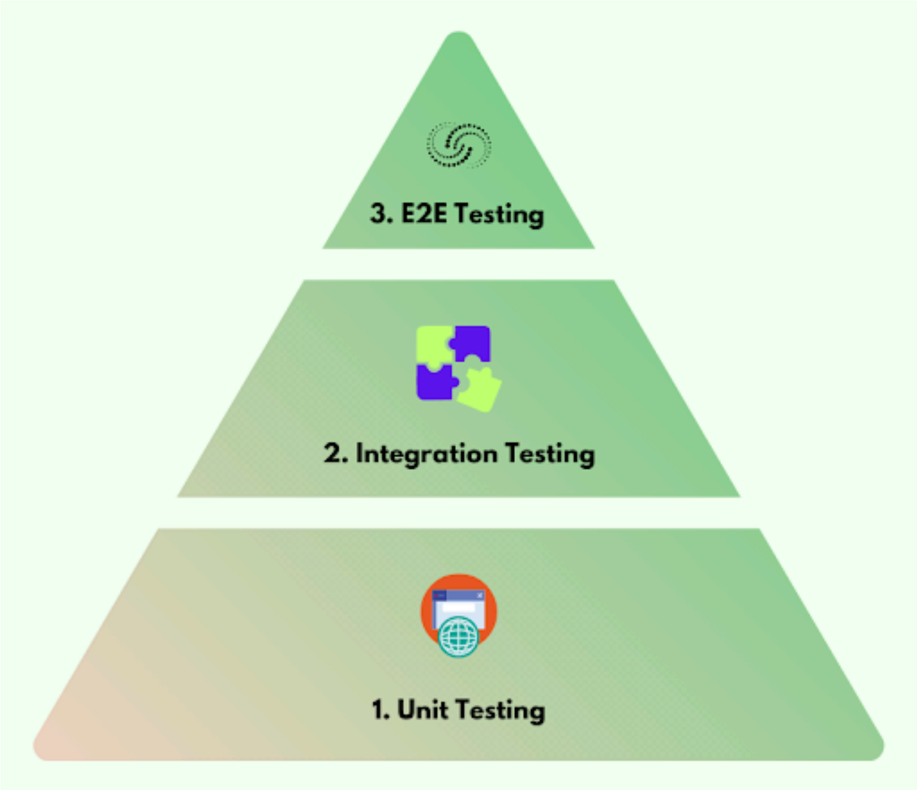


Build Trust in Your Code

“If it’s not tested, it’s broken.”

UNIT TESTS VS. EVERYTHING ELSE

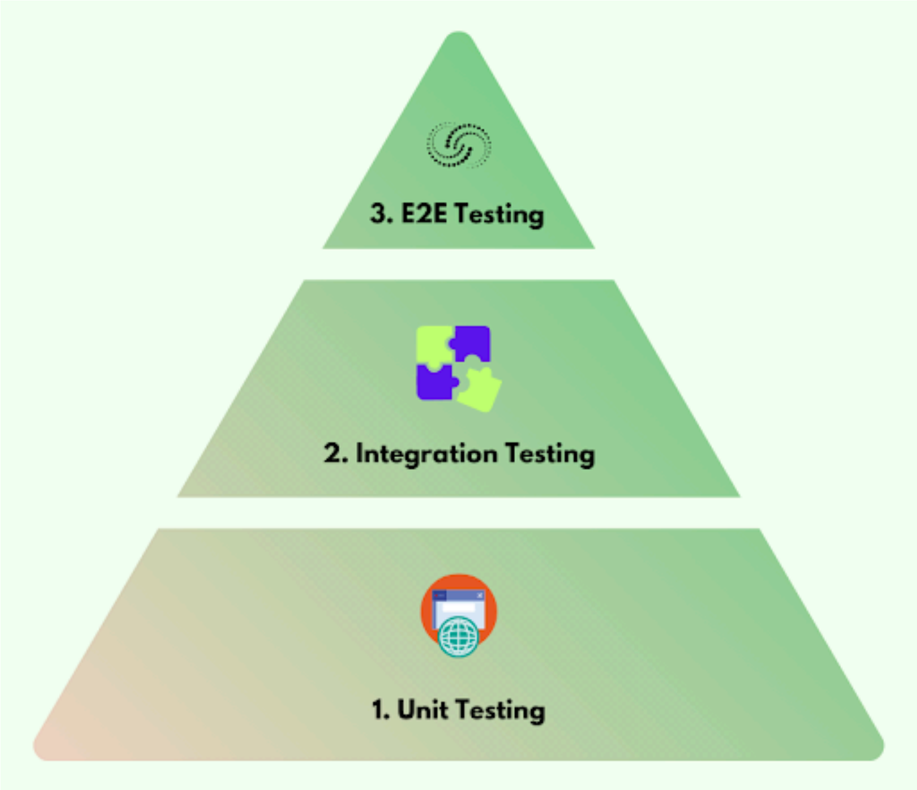
- Unit Tests (Base of the Pyramid)
  - Test the smallest pieces of code (e.g., functions, methods)
  - Fast, isolated, easy to write and run often



TestSigma

UNIT TESTS VS. EVERYTHING ELSE

- Integration Tests
  - check how modules work together (to come in iteration 2)

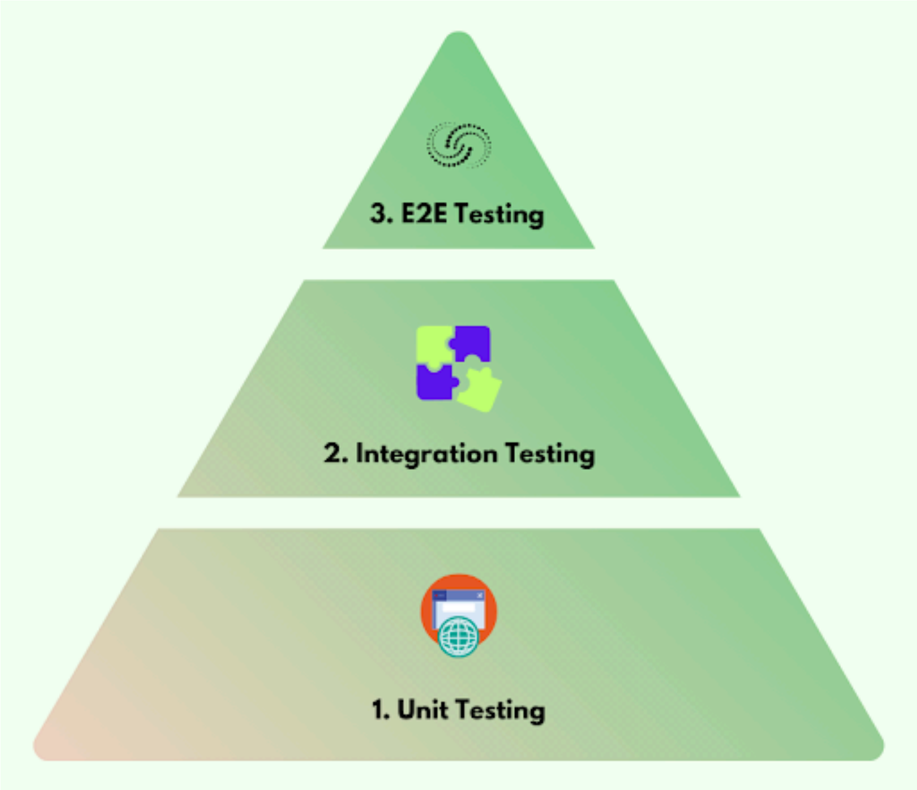


TestSigma

UNIT TESTS VS. EVERYTHING ELSE

- System Tests
  - test the full application as a whole
- End-to-End (E2E) Tests
  - simulate real user behavior across the stack
- Acceptance Tests
  - verify the app meets business/user requirements

Iteration 3!



TestSigma

ANATOMY OF A UNIT TEST

- Three Key Steps:
  - **Arrange** – Set up the data and environment
  - **Act** – Call the function or method you're testing
  - **Assert** – Check that the result matches your expectations

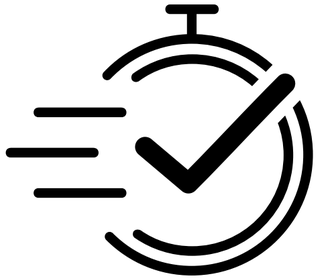
Let’s consider a function calculate_percentage(earned_points, total_points)

ANATOMY OF A UNIT TEST

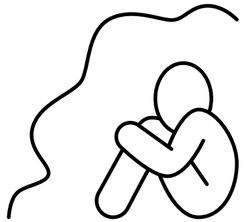
- Three Key Steps:
  - **Arrange** – Set up the data and environment
  - **Act** – Call the function or method you're testing
  - **Assert** – Check that the result matches your expectations

```python
def test_calculate_percentage():
    # Arrange
    total_points = 50
    earned_points = 45

    # Act
    result = calculate_percentage(earned_points, total_points)

    # Assert
    assert result == 90
```

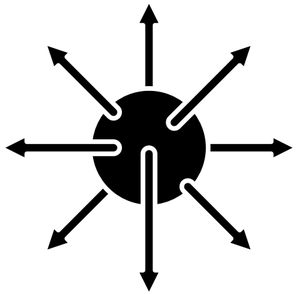
WHAT MAKES A GOOD UNIT TEST?



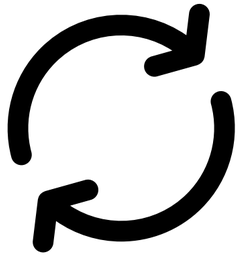
Fast



Isolated



Deterministic



Readable &
Repeatable

WHERE DO TESTS LIVE?

Project Structure:

- Production code goes in */src*
- Tests go in */test* (or sometimes */tests*)

Naming Conventions:

- Match test file names to the class or module being tested
- Use the format: *ClassNameTest* and *module_name_test*

Why This Matters:

- Makes it easy to find and maintain tests
- Many test runners auto-discover tests based on folder and file names

THE ROLE OF TESTS IN AGILE

- Fast Feedback is Key
  - Agile and CI/CD pipelines rely on quick test results to catch issues early
  - Automated tests run on every push to ensure stability and quality
  - (Remember we do this in SE2, not required for SE1)

THE ROLE OF TESTS IN AGILE

- Fast Feedback is Key
  - Agile and CI/CD pipelines rely on quick test results to catch issues early
  - Automated tests run on every push to ensure stability and quality
  - (Remember we do this in SE2, not required for SE1)

Importantly!

- Tests Make Refactoring Safe
  - Confidently change and improve code without breaking things
  - Acts as a safety net when your design evolves
- Connects to SOLID Principles **(coming soon!)**
  - Clean, modular code is easier to test — and good tests encourage clean design
  - Testing supports agility by reducing fear of change

PAUSE & REFLECT

Go find an old working Java program you have written in the last year or two.

Write a JUnit unit test for one of the methods in that program.

Does it follow Arrange → Act → Assert?