COMP 3550

8.4 — FLAKY & NON-DETERMINISTIC TEST STRATEGIES

Week 8: Advanced Testing

WHAT IS A FLAKY TEST?

Definition:

- A flaky test is a test that passes or fails unpredictably, even when the underlying code hasn’t changed.

Why It Matters:

- Breaks confidence in your test suite
- Wastes time in CI/CD pipelines
- Leads to “just rerun it” culture instead of fixing the issue

COMMON SOURCES OF NON-DETERMINISM

Cause	Example
Timing issues	Test fails if a delay or async result arrives late
Concurrency/race conditions	Threaded code behaves unpredictably during tests
External dependencies	Real network, DB, or file systems introduce delay or failure
Time/date reliance	Test passes today but fails tomorrow due to date logic
Randomness	Code includes Math.random() or shuffled order

STRATEGIES FOR STABILITY

1. **Replace sleep() with Event Signals or Polling**
   a. **instead of:** Thread.sleep(100); // hope it's ready!
   b. **do:** waitUntil(service::isReady, Duration.ofSeconds(2));

STRATEGIES FOR STABILITY

1. Replace sleep() with Event Signals or Polling
    1. **instead of:** Thread.sleep(100); // hope it's ready!
    2. **do:** waitUntil(service::isReady, Duration.ofSeconds(2));
2. **Inject Randomness Seeds for Reproducibility**
    1. If using any randomness: Random rng = new Random(42); // fixed seed
        1. Always log the seed if it’s dynamic
        2. Makes test failures reproducible when something goes wrong

STRATEGIES FOR STABILITY

1. Replace sleep() with Event Signals or Polling
   1. **instead of:** Thread.sleep(100); // hope it's ready!
   2. **do:** waitUntil(service::isReady, Duration.ofSeconds(2));
2. Inject Randomness Seeds for Reproducibility
   1. If using any randomness: Random rng = new Random(42); // fixed seed
      1. Always log the seed if it’s dynamic
      2. Makes test failures reproducible when something goes wrong
3. **Mock Time Instead of Relying on Real Time**
   1. Real time = unpredictable.
   2. **Use a Clock abstraction:** Clock testClock = Clock.fixed(Instant.now(), ZoneOffset.UTC);
   3. **Then inject it into your service:** new ExpiryService(testClock);

STRATEGIES FOR STABILITY

1. Replace sleep() with Event Signals or Polling
    1. **instead of:** Thread.sleep(100); // hope it's ready!
    2. **do:** waitUntil(service::isReady, Duration.ofSeconds(2));
2. Inject Randomness Seeds for Reproducibility
    1. If using any randomness: Random rng = new Random(42); // fixed seed
        1. Always log the seed if it’s dynamic
        2. Makes test failures reproducible when something goes wrong
3. Mock Time Instead of Relying on Real Time
    1. Real time = unpredictable.
    2. **Use a Clock abstraction:** Clock testClock = Clock.fixed(Instant.now(), ZoneOffset.UTC);
    3. **Then inject it into your service:** new ExpiryService(testClock);
4. **Replace External Services with Fakes or Mocks**
    1. Avoid relying on real network, file system, or DB
    2. Use in-memory fakes or fast mocks (as covered in 8.3)

STRATEGIES FO STABILITY

1. Replace sleep() with E nt Sig Pol
    a. **instead of:** Thread
    b. **do:** waitUr vi
2. Inject Random
    a. If usin
        i. At
        ii. Makes
3. Mock Tim
    a. Real time
    b. **Use a Cl** ZoneOffset.UTC);
    c. **Then inject i**);
4. **Replace Exter vice**
    a. Avoid relying on real DB
    b. Use in-memory fakes or fast cks (as ered in 8.3)

Stability comes from control. If your test depends on the environment, it's not really under test.

DETECTING FLAKY TESTS

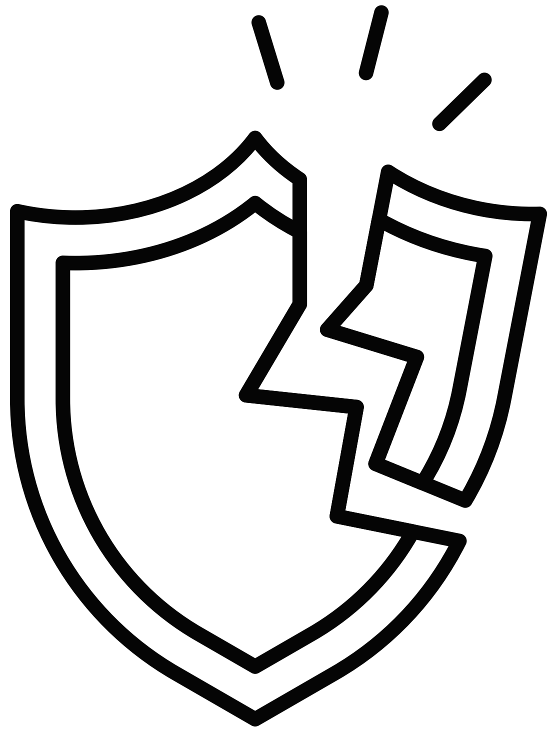
Watch for These Red Flags in CI (COMP 4550!):

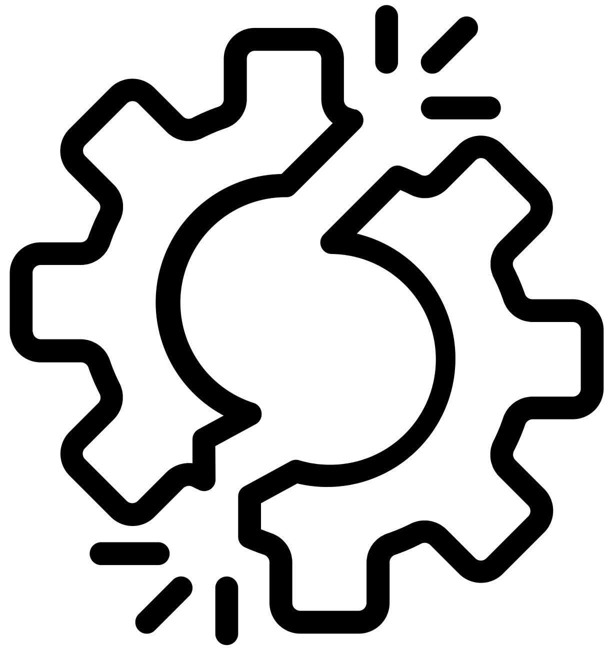
Symptom	Possible Cause
Test passes locally, fails in CI	Environment timing, resource conflicts
Test fails, then passes on rerun	Race condition or async delay
Test fails only when run in suite	Test order dependency or shared state

DETECTING FLAKY TESTS

Use CI Tools (COMP 4450!) & Logs (Now!):

- Look for intermittent failures over time
- Use test flakiness detection plugins if available (e.g., in Jenkins, GitHub Actions)
- Check if failures correlate with parallelism, resource contention, or network lag





PROJECT PAUSE & REFLECT

Audit your test suite.

Isolate one flaky test and try to stabilize it.