COURSE SYLLABUS

CS350 • Software Testing & Reliability

School of Computational Arts & Sciences • Graduate • 6 ECTS

Overview

Develop a practical testing and reliability toolkit: unit/integration strategies, property-based thinking, observability signals, and incident response. Students write tests that prevent regressions, design for failure, and produce postmortems that improve systems over time.

LOGISTICS

Credits: 6 ECTS
Level: Graduate
School: School of Computational Arts & Sciences
Prerequisites: None listed
Tags: testing, reliability, observability
Meeting time: Weekly lecture + reliability lab
Instruction mode: Test like you operate: observability, failures, postmortems

LEARNING OUTCOMES

You will be able to:
- Write effective tests at multiple levels (unit, integration, property-based)
- Design observability signals (logs, metrics, traces) that support debugging
- Run incident-style analysis and produce actionable postmortems
- Write effective tests at multiple levels (unit, integration, property-based)
- Design observability signals (logs, metrics, traces) that support debugging
- Run incident-style analysis and produce actionable postmortems

ASSESSMENT

Components
- Coursework: 60%
- Final project: 40%

Your grade reflects the quality of your testing strategy and your ability to reason about
failures. Projects include a reliability report: what can fail, how you detect it, and how
you recover.

WEEKLY PLAN

Schedule
Week 1: Week 1
  - Test design: boundaries, fixtures, and failure messages
Week 2: Week 2
  - Integration tests and contracts
Week 3: Week 3
  - Property-based thinking and fuzzing intuition
Week 4: Week 4
  - Observability: logs vs metrics vs traces

Extended outline
- Test design: boundaries, fixtures, and failure messages
- Integration tests and contracts
- Property-based thinking and fuzzing intuition
- Observability: logs vs metrics vs traces
- Incident response: postmortems and preventive actions
- Final: a small service with a testing + reliability pack

POLICIES & RESOURCES

- No flaky tests: deterministic assertions and controlled timeouts.
- Postmortems must be blame-free and specific.
- Academic integrity: you may share test ideas; submit your own code.

Suggested resources
- Testing checklist: arrange/act/assert, boundaries, invariants
- Postmortem template: timeline, root cause, action items
- Observability cheat sheet: choosing signals for questions