www.logicalimagination.com
800.657.1494

Testing, Debugging & Securing AI-Assisted Code

Course #: AI-302 **Duration:** 2 days

Prerequisites

Completion of AI Foundations for Software Developers and Professional Software Development with GitHub Copilot, or equivalent experience using AI-assisted coding tools in professional software development workflows.

Details

AI-assisted development accelerates code creation—but it also introduces new risks related to correctness, security, and long-term maintainability. This course focuses on the counter-skills developers need when working with AI-generated code: rigorous testing, disciplined debugging, and proactive security review.

Participants learn how to evaluate, test, debug, and secure code they did not write line-by-line, how to avoid false confidence from AI-generated tests and fixes, and how to maintain professional engineering standards in AI-augmented development environments.

After attending this course, students should be able to:

Identify common defects and risks in AI-generated code
Design effective tests for AI-assisted implementations
Debug AI-generated logic systematically and confidently
Recognize security vulnerabilities introduced by AI tools
Apply professional judgment to determine when AI output is safe to use

This course is designed for software developers and technical leads responsible for maintaining code quality, reliability, and security in environments where AI-assisted coding tools are in active use.

Software Needed

Participants must have a laptop or desktop computer (Windows, macOS, or Linux) with a modern code editor or IDE, reliable internet access, and a working local development environment for at least one commonly used programming language in their organization (for example, Python, JavaScript/TypeScript, Java, or C#). The environment must support running unit tests, stepping through code with a debugger, and installing dependencies (package manager access). Git and command-line access are required. If the organization uses AI-assisted coding tools (such as GitHub Copilot), participants should have access available for exercises, though the course can be completed using provided examples and standard tooling. All work should comply with organizational security, privacy, and confidentiality requirements.

Outline

Testing, Debugging & Securing AI-Assisted Code

- **Why AI-Assisted Code Requires Different Quality Controls**
  - How AI changes defect patterns and failure modes
  - Why “looks correct” is more dangerous than obvious bugs

    - The illusion of confidence in AI-generated output
    - Shifting responsibility in AI-augmented development

- **Evaluating AI-Generated Code**
    - Reading AI-generated code critically
    - Identifying hidden assumptions and missing constraints
    - Recognizing over-generalized or incomplete logic
    - Knowing when to refactor vs. rewrite

- **Testing AI-Assisted Code**
    - Why AI-generated code needs more testing, not less
    - Writing unit tests for AI-generated implementations
    - Designing tests for edge cases and failure scenarios
    - Avoiding shallow or misleading test coverage

- **AI-Generated Tests: Benefits and Risks**
    - When AI-generated tests are useful
    - Common gaps in AI-generated test suites
    - False confidence from passing tests
    - Improving AI-generated tests through human review

- **Debugging Code You Didn't Write**
    - Debugging strategies for unfamiliar logic
    - Tracing behavior back to incorrect assumptions
    - Identifying subtle behavioral regressions
    - Using AI tools to assist debugging responsibly

- **Fix-Forward Workflows**
    - Using tests to drive corrections
    - Avoiding infinite fix-prompt-fix cycles
    - Ensuring fixes don't introduce new defects
    - Maintaining clarity and intent during iteration

- **Security Risks in AI-Assisted Development**
    - Common insecure patterns introduced by AI
    - Input validation and injection vulnerabilities
    - Authentication and authorization mistakes
    - Dependency and configuration risks

- **Secure Review of AI-Generated Code**
    - Treating AI output as untrusted input
    - Incorporating security review into workflows
    - Knowing when human security expertise is required
    - Reducing risk without blocking productivity

- **Licensing, Compliance, and IP Considerations**
    - Understanding licensing risks in AI-generated code
    - Organizational policies and acceptable use
    - Documenting AI assistance where required
    - Aligning with legal and compliance expectations

- **Maintaining Quality at Scale**
    - Preventing AI-accelerated technical debt
    - Establishing quality standards for AI-assisted code
    - Supporting consistent practices across teams
    - Measuring quality alongside productivity