APPLIED TECHNOLOGY ACADEMY

800.674.3550
Info@AppliedTechAc.com
AppliedTechnologyAcademy.com

Software Engineering with Coding Agents: From Requirements to Verified Delivery

Applied Technology Academy

Turn coding-agent adoption into a repeatable engineering practice. Over three days developers move from an ambiguous software request to a tested, reviewed application increment while keeping control of requirements, architecture, security and delivery decisions. Participants use a coding agent in Visual Studio Code to extend a working business application, connect tools and context through the Model Context Protocol (MCP), configure a custom QA and security reviewer, and author reusable skills and deterministic tools. About half the course is hands-on practice.

LEVEL	DURATION	DELIVERY
Intermediate	**3 Days**	**Instructor-led**

Course Overview

- One application throughout: Project Northstar, an equipment reservation portal, supplies a
- continuous business context for requirements, implementation, debugging, security review and delivery.
- Approximately 50% hands-on practice, guided by an instructor.
- Vendor-neutral: the course covers Claude Code, GitHub Copilot and Codex, and each participant
- performs the labs with one selected agent platform.
- Three implementation tracks - Python/Flask (pytest, pytest-bdd), Java/Spring Boot MVC (Maven,
- JUnit, Cucumber-JVM) or C#/ASP.NET Core 10 (.NET SDK, xUnit, Reqnroll).
- Every task ends in independent proof: the agent's completion statement is never treated as evidence.

Course Outline

- **Day 1: Direct the Work**
- **Module 1. From Assistant to Coding Agent**
  - The agent loop: model, instructions, tools, environment, actions and feedback.
  - Suitable tasks, unreliable assumptions and the boundaries for autonomy.
  - Hands-on: build, test and launch the baseline application, then verify a small repository task independently.
- **Module 2. Make the Requirement Agent-Ready**
  - Goal, Constraints, Context, Done When and Independent Proof applied to a business request.
  - Example Mapping to uncover rules, boundary cases, open questions and nonfunctional requirements.

800.674.3550
Info@AppliedTechAc.com
AppliedTechnologyAcademy.com

  - Hands-on: convert an ambiguous reservation request into an approved task contract and example map.
- **Module 3. Build the Agent's Project Context**
  - Repository structure, entry points, dependencies and sources of truth.
  - Project and global instructions, scope and precedence, context limits and session handoffs.
  - Hands-on: produce a repository map and a minimal AGENTS.md, then repeat the baseline task and compare.
- **Module 4. Plan with Tests Before Implementation**
  - Given/When/Then scenarios and focused behavioral tests from approved examples.
  - Acceptance test-driven development; telling a missing capability from an environment failure.
  - Hands-on: capture expected failing results and approve a bounded implementation plan.
- **Day 2: Build, Debug, Review and Reuse**
- **Module 5. Implement in Controlled Slices**
  - Red-green-refactor, narrow diffs and Git checkpoints for recovery.
  - Focused checks before regression runs; model and task size chosen with quality, limits and cost in mind.
- **Module 6. Debug from Evidence**
  - Reproduce failures from real inputs, outputs, logs and environment details.
  - Falsifiable hypotheses, bounded experiments and stopping unproductive retry loops.
  - Hands-on: diagnose a seeded defect, explain the root cause and prove the repair with a regression test.
- **Module 7. Review Agent-Generated Code**
  - Requirements, final diff and test evidence reviewed in a fresh context.
  - Correctness, design fit, input validation, authorization boundaries, dependency provenance and licensing.
  - Weak assertions, missing negative cases and unsupported completion claims.
- **Module 8. Build Custom Skills and Tools from the Proven Workflow**
  - What belongs in a prompt, an instruction, a skill, a script or a tool.
  - Packaging a skill with a clear trigger, bounded steps, dependencies, expected output and proof method.
  - Building a deterministic verification or evidence-report tool, and versioning agent assets.
- **Day 3: Implement a New Requirement Using the Established Workflow**
- **Module 9. Integrate MCP Tools and Context**
  - How MCP connects agents to tools and context, and retrieved data versus authorized instructions.
  - Least privilege, permission controls, secret handling, provenance and prompt injection.

800.674.3550
Info@AppliedTechAc.com
AppliedTechnologyAcademy.com

  - Hands-on: connect a read-only MCP server, verify its access boundary and reject an embedded instruction.
- **Module 10. Build a Custom QA and Security Reviewer Agent**
  - Reusable skills versus specialist roles, agent definitions and review handoffs.
  - Bounded context, restricted tools and verified permissions; alternate models and shared blind spots.
- **Module 11. Continuous Verification and Incremental Delivery**
  - Tests, static checks, dependency and security checks and requirement-to-evidence traceability in one gate.
  - Mapping the gate to a hosted CI/CD workflow: triggers, failure handling, artifacts and approval points.
  - Delivery handoff with known limitations, rollback considerations and outstanding operational checks.
- **Module 12. Capstone: Implement and Verify the New Requirement**
  - Carry a new requirement through the established contract, tests, implementation, review and verification path.
  - Reuse the project instructions, custom skill and custom tool; invoke the custom reviewer.
  - Demonstrate the application, explain the evidence and obtain a human acceptance decision.

Prerequisites

Participants should have:

- working knowledge of Python, Java or C#, including the ability to read and modify a small web
- application in the selected track
- basic familiarity with Git, automated tests, terminal commands and Visual Studio Code
- a development computer with permission to install or use the selected runtime, VS Code
- extensions and lab dependencies
- organization-approved access to a supported coding agent, with the capabilities and usage
- allowance the selected configuration requires
- network access to the selected agent service and required dependency sources
- No prior experience with coding agents, agent skills or MCP is required. Setup guides, command
- references, checkpoints and recovery instructions support the common lab path, and optional
- extension challenges accommodate more experienced participants.

Who Should Attend

- Software developers and technical leads who write or review application code.
- Teams beginning to use coding agents.

- Teams that want more consistent results from adoption already under way.

What You'll Learn

By the end of this course, participants will be able to:

- distinguish models, chat assistants, coding agents, instructions, skills, tools and specialist agents
- define an agent task using Goal, Constraints, Context, Done When and Independent Proof
- orient an agent to an unfamiliar repository and establish durable project instructions
- turn business examples into executable acceptance criteria and tests that prove missing behavior
- implement, debug and refactor a bounded change using incremental verification and human review
- evaluate generated code for correctness, maintainability, security, dependency risk and scope
- connect an MCP server for tools and external context with explicit trust boundaries
- configure a custom QA and security reviewer agent and reconcile its findings with tests
- author and evaluate custom skills and deterministic tools for reuse at project or global scope
- produce a verified increment, an evidence-based delivery handoff and an improvement plan

Hands-On Work

Representative labs:

- extend a working reservation portal with date validation and availability rules
- diagnose and correct a seeded defect, then convert it into a regression check
- review a deliberately flawed change and produce an evidence-backed disposition
- connect a read-only MCP server and verify its access boundary
- author a custom skill that invokes a deterministic tool, and validate both from a clean repository
- Deliverables participants keep: task-contract and review templates, project instructions, a tested
- custom skill and tool, an MCP connection configuration, a reviewer-agent configuration, verification
- commands and a delivery evidence package.

Environment and Tools

- Visual Studio Code with the selected coding agent (Claude Code, GitHub Copilot or Codex).
- One lab track: Python/Flask, Java/Spring Boot MVC or C#/ASP.NET Core 10.
- A supplied baseline application with seeded data, a passing test suite and a clean Git baseline.
- A read-only MCP server supplied for the Day 3 maintenance-rules exercise.
- A pre-course environment check confirms runtime versions, extensions, account access,
- organization policies and the agent capabilities the labs require.

800.674.3550
Info@AppliedTechAc.com
AppliedTechnologyAcademy.com

Next Steps

- Participants leave with a workplace use case, an owner for shared agent assets, evaluation
- fixtures and the conditions that trigger revalidation - the basis for a maintainable practice in
- which developers remain accountable for the software and deliberately improve the agents they use.

Ready to enroll? Call 800.674.3550 or email Info@AppliedTechAc.com — private team cohorts, GSA MAS purchasing, military credentialing funding, VA GI Bill and VR&E and student financing available.