APPLIED AI ENGINEERING ACADEMY

Agentic Engineering

From Tool Calling and MCP to Governed, Testable Agent Systems

3 DAYS | 21 CONTACT HOURS | HANDS-ON INTENSIVE

A software-engineering course for moving beyond prompt demos and vibe-coded prototypes toward typed tools, durable workflows, security controls, evaluations and auditable governance.

AUDIENCE	DELIVERY	CAPSTONE
AI and software engineers, architects, security specialists and technical product owners.	One agent is progressively extended with tools, MCP, state, approvals, tests, security and governance evidence.	A governed research or operations agent with typed contracts, an MCP server, evaluation and red-team evidence.

Feasibility verdict: Feasible when one primary Python stack is used and multi-agent systems remain a design exercise. Building several unrelated agents or comparing every framework hands-on would not fit.

Version 1.0 | Framework review date: 1 August 2026

Course overview

The course teaches the engineering layer around language models: harnesses, tools, reusable skills, MCP interoperability, state, durable execution, validation, security, observability and governance. Participants transform a functioning but fragile agent into a controlled application.

Learning outcomes

- Distinguish chatbots, deterministic workflows, tool-using agents and multi-agent systems and select the simplest suitable architecture.
- Explain the responsibilities of an agent harness: instructions, context, tools, state, control loop, permissions, budgets, validation and observability.
- Build typed tools with preconditions, postconditions, invariants, explicit errors and approval requirements.
- Package a reusable skill and explain progressive disclosure, versioning, provenance and skill security.
- Build and consume an MCP server using the current stateless 2026-07-28 specification concepts for tools, resources and prompts.
- Add state, checkpoints, retries, idempotency, human-in-the-loop gates and safe termination to an agent workflow.
- Move beyond vibe coding through specifications, tests, static analysis, CI, sandboxes, dependency controls and reviewable changes.
- Evaluate complete trajectories and map technical evidence to NIST AI RMF, ISO/IEC 42001, the EU AI Act and OWASP agentic risks.

Realistic depth within three days

IMPLEMENT	PRACTISE	SURVEY / DEMO
• Typed agent and tool contracts • Local MCP server and client • Evaluation and approval workflow	• State, checkpoints and idempotency • Security tests and trajectory analysis • Governance and operational evidence	• Large multi-agent platforms • Enterprise certification programmes • Provider-specific agent products and deployment clouds

Conditions that make the schedule feasible

- Use one capstone codebase for all three days; every new concept improves the same system.
- Use PydanticAI and the official MCP/FastMCP tooling for core labs, with LangGraph introduced only for explicit durable orchestration.
- Treat OpenAI Agents SDK and Microsoft Agent Framework as framework comparisons, not additional participant implementations.
- Use mock tools and synthetic data; no workshop agent receives production credentials or unrestricted external side effects.
- Teach one multi-agent pattern as a paper or whiteboard design exercise. Single-agent reliability is the practical priority.
- Provide a deliberately fragile starter agent so participants can focus on engineering controls rather than UI and boilerplate.

Recommended participant profile

Primary audience. Software and AI engineers, solution architects, technical product owners, platform engineers, security specialists and governance practitioners who build or review agentic applications.

Prerequisites. Python, APIs, JSON schemas, Git and basic LLM prompting. Familiarity with asynchronous code and testing is helpful. Course 1 or equivalent LLM application experience is recommended.

Day 1 - Harnesses, Tool Contracts, Skills and MCP

Build a constrained single agent before adding orchestration complexity.

Time	Module	Content and activity
09:00-09:30	**Architecture baseline**	Chatbot, workflow and agent distinctions; autonomy spectrum; when conventional software is the better answer.
09:30-10:30	**Harness engineering**	Model, instructions, context, tools, state, control loop, policies, budgets, stop conditions, errors and observations.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**Tool calling and contracts**	JSON schemas, typed dependencies, preconditions, postconditions, invariants, error types, timeouts, retries and result validation.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:00	**Permissions and approvals**	Read versus write tools, least privilege, capability grants, confirmation, compensation and side-effect boundaries.
14:00-15:00	**Skills**	Reusable procedures, manifests, supporting files, progressive loading, versioning, provenance and malicious-skill risks.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**MCP architecture**	Hosts, clients and servers; tools, resources and prompts; current stateless request model, metadata, caching and authorization concerns.
16:00-17:30	**Lab 1: typed agent and MCP**	Build a small typed agent, create read-only and side-effecting tools, expose a local MCP server and require approval for writes.

DAY OUTPUT

A constrained single agent with typed tools, explicit errors, a local MCP server, a permission matrix and an approval gate.

INSTRUCTOR NOTE

The current MCP 2026-07-28 specification introduced breaking stateless-core changes. Pin the SDK and teach protocol concepts, not memorization of transport boilerplate.

Day 2 - Durable Orchestration and Beyond Vibe Coding

Make execution explicit, resumable and maintainable.

Time	Module	Content and activity
09:00-09:30	**Recap and trace review**	Inspect Day 1 traces and identify ambiguous tool descriptions, malformed arguments and unsafe side effects.
09:30-10:30	**Agent and workflow patterns**	ReAct, router, planner-executor, evaluator-optimizer, supervisor-worker and deterministic workflow boundaries.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**State, context and memory**	Conversation state, workflow state, long-term memory, context selection, data lifetimes, poisoning and deletion policies.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:15	**Durable execution**	Checkpoints, resumability, retries, idempotency, compensation, human interrupts and failure recovery.
14:15-15:00	**Multi-agent trade-offs**	Specialization, handoffs, shared state, communication overhead, coordination failure and when not to use multiple agents.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**Beyond vibe coding**	Requirements, repository structure, architecture decisions, unit and integration tests, static analysis, CI, dependency pinning and sandboxed execution.
16:00-17:30	**Lab 2: productionize the agent**	Add explicit state, checkpoints, idempotency, retries, human review and tests to a deliberately fragile starter implementation.

DAY OUTPUT

A restartable, testable agent workflow with explicit state, failure handling, review gates and a maintainable repository structure.

INSTRUCTOR NOTE

Do not build a multi-agent system in parallel with the main lab. A single-agent implementation plus a multi-agent architecture exercise gives better learning and fewer hidden failure modes.

Day 3 - Security, Evaluation, Governance and Deployment

Prove what the agent can do, constrain what it may do and document how it is governed.

Time	Module	Content and activity
09:00-10:00	**Threat modelling**	Assets, identities, trust boundaries and data flows; goal hijacking, prompt injection, tool misuse, privilege abuse and memory poisoning.
10:00-10:30	**Guardrails and validation**	Input, tool-call, tool-result, output and trajectory validation; sandboxing and safe failure.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**Agent evaluation lab**	Task success, tool selection, argument correctness, side-effect correctness, trajectory efficiency, robustness and adversarial cases.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:15	**Governance evidence**	System inventory, intended purpose, ownership, risk classification, approvals, monitoring, changes, incidents and human oversight.
14:15-15:00	**Standards and regulation**	NIST AI RMF, ISO/IEC 42001 and 23894, EU AI Act responsibilities and OWASP Top 10 for Agentic Applications.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**Operations and deployment**	Secrets, workload identity, network policy, observability, budgets, model routing, rollback and incident response.
16:00-17:00	**Capstone red team**	Attack another team's agent, remediate findings and finish the technical and governance evidence package.
17:00-17:30	**Demonstrations and review**	Demonstrate controlled behaviour, evidence, unresolved risks and a deployment recommendation.

DAY OUTPUT

A governed agent prototype with evaluation evidence, security tests, traces, risk register, tool inventory and deployment recommendation.

INSTRUCTOR NOTE

Governance work produces an evidence package, not legal advice or ISO certification. Current EU AI Act dates and framework guidance should be revalidated before delivery.

Hands-on structure and capstone

Progressive lab sequence

Lab	Focus	Participant deliverable
1	Contracts and MCP	Typed agent, safe tools, local MCP server, permission matrix and human approval.
2	Durable orchestration	Explicit state, checkpoints, retries, idempotency, review gates and repository tests.
3	Evaluation and red team	Trajectory dataset, task and tool metrics, adversarial cases and remediation evidence.
4	Governed capstone	Operational agent, risk register, tool/data inventory, trace evidence and deployment decision.

Capstone specification

Teams build a governed research or operations agent against synthetic services. The same agent is extended across all three days so participants experience the full path from a working demo to a controlled system.

- Documented intended purpose, users, exclusions and autonomy boundary.
- At least three typed tools with explicit errors and separate read/write permissions.
- At least one reusable skill and one MCP server or MCP-exposed capability.
- State, maximum-step and cost budgets, safe termination and resumability.
- Human approval for sensitive actions and idempotency for side effects.
- A minimum 20-case trajectory evaluation set with normal, failure and adversarial scenarios.
- Trace collection, tool and data inventory, risk register and incident/rollback procedure.
- A deployment recommendation: approve, approve with conditions or do not deploy.

Assessment

Component	Weight
Concept checks and architecture decisions	15%
Three progressive engineering labs	35%
Governed capstone and evidence package	35%
Red-team defence and deployment recommendation	15%

Framework stack and delivery operations

Primary workshop stack

Layer	Recommended framework	Why it is used in this course
Typed agent layer	PydanticAI	Typed dependencies, tool schemas, output validation, approvals, MCP integration and evaluation-friendly design.
MCP implementation	Official MCP SDK / FastMCP	Builds a standards-based server and exposes protocol concepts directly.
Orchestration	LangGraph	Makes deterministic and model-driven steps explicit and supports persistence and human interrupts.
Observability	OpenTelemetry-compatible tracing	Vendor-neutral traces for models, tools, workflows, latency, costs and errors.
Testing	pytest + typed fixtures	Validates deterministic code, contracts, tool results and regression scenarios.
Security and policy	Sandbox + explicit allow-lists	Separates model decisions from credentials, network access and side effects.

Frameworks discussed but not all taught hands-on

- OpenAI Agents SDK is discussed for managed turns, tools, handoffs, guardrails, sessions and built-in tracing.
- Microsoft Agent Framework is discussed for enterprise workflows, typed executors, state management and explicit multi-agent orchestration.
- PydanticAI graph and durable-execution integrations can replace LangGraph in teams that prefer a single framework; the key learning objective is explicit state and recovery semantics.
- Crew-style multi-agent frameworks are not the workshop default because coordination complexity can obscure contracts, security and evaluation.

Operational requirements

PARTICIPANT SETUP

Laptop with Python 3.11+, Git, Docker or a supplied local sandbox, and the pinned workshop repository. All external services are mocked or use workshop-scoped credentials.

INSTRUCTOR PREPARATION

Pin the newly updated MCP SDK, test local transports and approval flow, provide a fragile starter agent, a mock service suite, attack cases and completed fallback traces.

CLASS SIZE

12-20 participants per instructor. Add a teaching assistant above 16 or when participants have limited software-testing experience.

DATA AND SAFETY

Use synthetic identities, documents and services. Never provide unrestricted shell, production cloud, personal mailbox, calendar, payment or database credentials in a classroom agent.

Framework versions should be pinned and the complete lab environment should be tested immediately before delivery. The framework recommendations reflect documentation available on 1 August 2026.

Selected current references

Official documentation and standards used to validate the framework recommendations and the current syllabus scope. Accessed 1 August 2026.

1. MCP 2026-07-28 specification — *Model Context Protocol.* Current stateless-core specification, tools, resources, prompts and protocol metadata.
2. PydanticAI documentation — *Pydantic.* Typed agents, validation, approvals, MCP, durable execution and graph support.
3. LangGraph documentation — *LangChain.* Durable execution, persistence, interrupts and explicit graph orchestration.
4. OpenAI Agents SDK — *OpenAI.* Managed turns, tools, handoffs, guardrails, sessions and tracing.
5. Microsoft Agent Framework — *Microsoft.* Typed agents, enterprise middleware, workflows and state management.
6. AI Risk Management Framework — *NIST.* Voluntary framework and generative-AI profile for managing AI risks.
7. ISO/IEC 42001 — *ISO.* Requirements for an organizational AI management system.
8. OWASP Top 10 for Agentic Applications 2026 — *OWASP GenAI Security Project.* Current agentic threat categories and mitigation guidance.
9. EU AI Act implementation overview — *European Commission.* Current application timeline and responsibilities for providers and deployers.

Syllabus maintenance note

Before each delivery, verify the MCP specification and SDK version, framework APIs, agent-security guidance, AI Act application dates and any changes to provider tool, skill, session and tracing semantics.

> **Important:** The workshop produces a governed prototype and evidence package. It does not constitute legal advice, ISO/IEC 42001 certification, a penetration test or approval for autonomous production operation.