

ENTERPRISE PROGRAM BRIEF

Agentic AI Engineering

Develop a bounded AI agent that retrieves approved knowledge, invokes validated tools, requests human approval for consequential actions, and produces evaluation evidence. Participants work through architecture, context, tool contracts, workflow orchestration, testing, and deployment planning using one consistent implementation path.

VNode ITeS

VAI-201

Coming soon

DURATION
40 hours

LEVEL
Intermediate

FORMAT
Private cohort

AUDIENCE PROFILE

Built for these roles

Software developers, AI engineers, solution architects, and experienced technical consultants.

OVERVIEW

Executive overview

Develop a bounded AI agent that retrieves approved knowledge, invokes validated tools, requests human approval for consequential actions, and produces evaluation evidence. Participants work through architecture, context, tool contracts, workflow orchestration, testing, and deployment planning using one consistent implementation path.

READINESS

Prerequisites

- Working Python, REST APIs, JSON, Git, basic authentication/authorisation, and introductory LLM and retrieval concepts. Participants should be able to debug a small API application.

PROGRAM OUTCOMES

Capabilities your teams will gain

- Decide when a deterministic workflow is preferable to agent planning.
- Build an agent with explicit context and tool contracts.
- Enforce action authority outside generated text.
- Evaluate tool selection, task completion, and failure behaviour.
- Prepare a versioned deployment candidate and explain its limits.

ENTERPRISE CUSTOMIZATION

Enterprise customization

Tailor this program to your organization's priorities: Build tool-using AI applications with controlled actions and measurable behaviour

- 5 learning days of 8 hours, or 10 sessions of 4 hours. Learning hours exclude breaks and include module exercises and capstone time. Setup is completed before the course.

CURRICULUM

Curriculum roadmap

1 MODULE 1 · 2 PROPOSED HOURS

Agent architecture and task boundaries

An agent combines model decisions with an execution environment. Begin with a task that needs flexible interpretation, then define a stopping condition and authority boundary. A deterministic function remains the better choice for an explicit stable rule.

- Agent loop versus deterministic workflow
- Task scope, success criteria, and allowed actions
- Single-agent baseline before additional agents

Planned lab: Compare three requests and design a bounded support triage task. Identify which steps remain ordinary code.

Expected output: Task contract and architecture decision.

2 MODULE 2 · 4 PROPOSED HOURS

Model interfaces and structured outputs

Treat model output as input to a software boundary. Validate schemas, required fields, and allowable values before the application acts. Separate the model deployment identifier from an assumed public model name, and keep credentials out of prompts and source control.

- Messages and model adapters
- Typed result schemas and validation
- Configuration, secrets, and token limits

Planned lab: Connect one approved deployment, request a structured triage result, and reject malformed or unsupported values. Use recorded fixtures when live access is unavailable and label them accordingly.

Expected output: Validated model adapter and output contract.

3 MODULE 3 · 6 PROPOSED HOURS

Context and retrieval engineering

The information supplied to the model shapes the answer. Preserve source identity and access metadata as content is prepared. Compare retrieval candidates with expected evidence rather than assuming that a fluent answer demonstrates correct retrieval.

- Source selection, metadata, and freshness
- Chunking and retrieval experiments
- Context budget and source attribution

Planned lab: Ingest a small synthetic corpus, compare two retrieval settings, and show that restricted records are excluded before generation.

Expected output: Versioned corpus, retrieval comparison, and access tests.

4 MODULE 4 · 6 PROPOSED HOURS

Tools and MCP integration

Tool definitions expose application capabilities to an agent, but a schema does not grant permission. The host must validate identity, arguments, scope, and execution results. MCP is an integration protocol, not a replacement for authentication or business approval.

- Tool schemas and server-side argument validation
- MCP host client and server responsibilities
- Least privilege, timeouts, and safe tool responses

Planned lab: Expose a read-only synthetic order lookup through a local MCP server and call it from the agent. Add negative argument and permission cases.

Expected output: Tool server, adapter, and boundary checks.

5 MODULE 5 · 6 PROPOSED HOURS

Workflow orchestration and human approval

Use explicit workflow structure when execution order or approvals matter. Maintain state across steps so an interrupted operation can resume without silently repeating side effects. Human approval must be tied to the exact proposed action and authenticated reviewer.

- Sequential and conditional execution

- Workflow state and bounded retries
- Approval context, expiry, and duplicate protection

Planned lab: Build a classify-retrieve-draft-review workflow. Pause before a ticket write, then test denied, approved, and repeated submissions.

Expected output: Stateful workflow and approval evidence.

6

MODULE 6 · 4 PROPOSED HOURS

Agent evaluation and failure analysis

Measure whether the agent completed the intended task and used tools correctly. Keep development examples separate from holdout cases. Inspect intermediate events to distinguish retrieval errors, planning errors, bad arguments, and unsuitable source content.

- Task and tool-call evaluation
- Holdout cases and human review
- Regression analysis and trace inspection

Planned lab: Run a small test suite containing normal tasks, no-evidence requests, incorrect tool arguments, and dependency failures. Review a failed trace.

Expected output: Evaluation report and prioritised defect list.

7

MODULE 7 · 4 PROPOSED HOURS

Operational and security controls

A deployment candidate needs reproducible configuration, observable behaviour, and limits on spending and execution. Platform controls should contain the consequences of unexpected model behaviour. Treat prompt injection as an ongoing engineering risk rather than a problem solved by one instruction.

- Execution budgets and sensitive-data handling
- Telemetry and versioned configuration
- Release evidence, rollback, and incident ownership

Planned lab: Inject a controlled tool failure, inspect metadata-only telemetry, and rehearse reverting an application or prompt version.

Expected output: Runbook and deployment candidate checklist.

8

MODULE 8 · 8 PROPOSED HOURS

Integrated capstone

Combine the previous modules into one demonstrable agent. Reviewers should be able to trace a task from user input through permitted evidence and tool calls to the final result. Explain unresolved limitations and the work needed for a real enterprise rollout.

- Integration and acceptance review
- Demonstration of normal and blocked actions
- Engineering handover and next-stage proposal

Planned lab: Deliver a service-operations agent with one read tool, one explicitly reviewed write, restricted evidence, and a test report.

Expected output: Working capstone, repository, evaluation evidence, and handover.

CAPSTONE

Capstone

A support triage agent that retrieves approved guidance, reads a synthetic service record, prepares an escalation, and submits only after a separate authorised approval.

ASSESSMENT

Assessment

Task and tool correctness 30%; context and access boundaries 25%; engineering and evaluation evidence 30%; handover 15%. Proposed pass: 70% overall with no unresolved critical access or unapproved-action failure.

DELIVERABLES

Participant deliverables

- Agent architecture and task contract
- Versioned code and source corpus
- Tool contracts and approval workflow
- Evaluation report and operating notes

SCOPE

Scope boundaries

Foundation-model training, advanced reinforcement learning, a survey of multiple frameworks, GPU infrastructure, and live writes to customer systems. ****Delivery options:**** 5 learning days of 8 hours, or 10 sessions of 4 hours. Learning hours exclude breaks and include module exercises and capstone time. Setup is completed before the course.

DELIVERY

Delivery options

5 learning days of 8 hours, or 10 sessions of 4 hours. Learning hours exclude breaks and include module exercises and capstone time. Setup is completed before the course. Proposed hours exclude breaks and are subject to pilot validation.

PRIVATE DELIVERY

Plan a private cohort for your team

Scope delivery format, role mix, and rollout timeline with our solutions team.

[Plan Private Delivery](#)
[Book advisory call](#)