AI Capability Programme: AI Engineering

Duration: 6 Weeks (81 Hours) | Instructor-led (Online)

License / Access Requirements

Paid

- **Azure subscription**
- **LLM API quota sized for a cohort of approximately 25 learners**
- **Approved AI assistant licences (Copilot Chat, ChatPwC)**

Free / open-source

- Python, VS Code, Git, Docker
- Open-source RAG, agent and evaluation frameworks (to be selected)

Environment

- Per-learner sandbox on the assigned cloud, with budget alerts and a hard cost cap
- Cloud identity and access provisioned before Week 1, including the services used in Week 5
- Stable internet connection and a second screen recommended for live coding

Course Objective

This OneCloud programme builds the capability to engineer reliable, governed LLM and agent-based systems on the cloud. Participants gain hands-on expertise in LLM application development, retrieval-augmented generation (RAG), retrieval engineering and evaluation, and AI agents with tool integration, memory and human checkpoints. They also become proficient with agent evaluation, security and governance, learn to deploy on the assigned cloud, and apply cost optimisation to keep systems within budget.

Each week follows the same rhythm: four 1.5-hour masterclasses (Monday to Thursday), a 1.5-hour reflection (Friday) and two 3-hour hands-on blocks (Saturday). A team hackathon in Week 3 and an individual capstone in Week 6 are assessed by a panel against a rubric, with Competent as the pass bar.

Target Audience

- Software engineers with three to eight years' experience
- Machine learning engineers moving into LLM and agent engineering
- Engineers building LLM, RAG and agent solutions on the cloud

Prerequisites

- Three to eight years of experience as a software or ML engineer
- Proficiency in Python and REST APIs
- Working knowledge of containers (Docker) and a cloud console
- Familiarity with Git and version control

Week-wise Course Content

Week 1: LLM Application Foundations, Prompting & Reliable API Integration

Course Outcomes

- Explain how LLMs work, including tokens, context windows and sampling, and their practical limits

- Select a model for a use case by comparing quality, latency, cost and licensing
- Apply prompting techniques (zero-shot, few-shot, chain-of-thought, system prompts) with structured JSON output
- Build an LLM service on a cloud-hosted model API with streaming, retries and timeout handling
- Implement function (tool) calling for simple actions
- Describe the RAG pattern and decide when retrieval is needed
- Track token usage and estimate cost per request

Hands-on Labs

- Block A: Compare two models on a banking FAQ question set
- Block A: Build a prompt chain with structured output and validation
- Block B: Build an LLM service with streaming, retries and usage logging
- Block B: Add a simple tool call to the service

Masterclasses (Monday to Thursday)

- How LLMs work and how to choose a model
- Prompting, system prompts and structured outputs
- LLM APIs and SDKs: streaming, retries, timeouts and rate limits
- The RAG pattern: when to retrieve and when not to

Tools / Demos: Assigned-cloud LLM service, Python, API client libraries, VS Code, Git

Week 2: RAG Pipelines, Embeddings & Retrieval Engineering

Course Outcomes

- Explain embeddings and vector similarity, and choose an embedding model and chunking strategy
- Build a RAG pipeline: ingestion, chunking, embedding, indexing, retrieval and generation
- Implement hybrid search with reranking and metadata filters
- Apply query rewriting and context assembly to improve retrieval
- Return citations that trace each answer to its source passages
- Design a safe fallback when evidence is missing or weak
- Handle document updates, access control and versioning in the index

Hands-on Labs

- Block A: Ingest and chunk insurance policy documents
- Block A: Build a vector index and baseline retrieval
- Block B: Add hybrid search, reranking and filters
- Block B: Return cited answers with an evidence-missing fallback

Masterclasses (Monday to Thursday)

- Embeddings, vector search and chunking strategies
- Hybrid search, reranking and metadata filtering
- Query transformation and context assembly
- Grounding and citations for regulated answers

Tools / Demos: Assigned-cloud LLM service, embedding model, vector or search index, reranker, Python

Week 3: Retrieval Evaluation, Agent Foundations & Hackathon

Course Outcomes

- Build a golden evaluation set and measure retrieval recall, precision and ranking quality
- Evaluate generated answers for groundedness and relevance, and explain the limits of LLM-as-judge
- Tune chunking, retrieval and prompts using evaluation results
- Describe the agent loop and add a tool call to a RAG assistant
- Deliver a team solution within a stated cost ceiling

- Record decisions and individual contributions in a team decision log

Hands-on Labs

- Block A: Build a golden set and score retrieval quality
- Block A: Score groundedness and tune the pipeline
- Block B: Hackathon: team build under a cost ceiling
- Block B: Demonstrate the build and submit the decision log

Masterclasses (Monday to Thursday)

- Evaluating retrieval: golden sets, recall and precision
- Evaluating generation: groundedness and LLM-as-judge
- Tool calling and the agent loop
- Hackathon briefing: brief, cost ceiling and team roles

Tools / Demos: Assigned-cloud LLM service, vector or search index, evaluation framework, agent framework, cost tracking

Week 4: AI Agents, Tool Integration, Memory, Evaluation & Security

Course Outcomes

- Design an agent architecture: planner, tools, memory and control loop
- Integrate tools through function calling and Model Context Protocol (MCP) servers
- Implement short-term and long-term memory and manage agent state
- Add human checkpoints (approval gates) for high-risk actions
- Describe long-running agent patterns: durable state, resumption and retries
- Evaluate agents using task success, tool-call accuracy and trajectory review
- Identify agent security risks (prompt injection, tool abuse, excessive agency) and apply least-privilege controls

Hands-on Labs

- Block A: Build a tool-using agent with MCP tools
- Block A: Add memory and a human approval checkpoint
- Block B: Create an agent evaluation harness
- Block B: Red-team the agent for prompt injection and tool abuse
- Block B: Apply least-privilege tool permissions

Masterclasses (Monday to Thursday)

- Agent architecture and tool integration (function calling and MCP)
- Memory, state, long-running agents and human checkpoints
- Evaluating agents: task success, tool use and trajectories
- Agent security and prompt injection defence

Tools / Demos: Agent framework, MCP servers, assigned-cloud LLM service, evaluation framework, Python

Week 5: AI Governance, Cloud Deployment & Cost Optimisation (Azure or Google Cloud)

Course Outcomes

- Explain the main obligations of the EU AI Act and the DPDP Act for AI systems at an overview level
- Map governance requirements to technical controls (logging, consent, data minimisation, human oversight)
- Assign agents their own identities and enforce least-privilege access
- Capture audit trails and evidence that a reviewer can inspect
- Deploy a RAG and agent system on the assigned cloud using managed AI services
- Configure secrets, networking and access for an AI workload on the cloud

- Apply AI FinOps: track cost per request, set budgets and reduce spend with caching, model routing and prompt trimming

Hands-on Labs

- Block A: Map governance requirements to controls for the system
- Block A: Add agent identity, audit logging and consent checks
- Block B: Deploy the system on the assigned cloud
- Block B: Set budgets, alerts and a cost cap
- Block B: Optimise cost with caching and model routing

Masterclasses (Monday to Thursday)

- AI governance: EU AI Act and DPDP Act obligations
- Agent identity, access control and audit trails
- AI architecture and deployment on the assigned cloud
- AI FinOps: cost drivers, budgets and optimisation

Tools / Demos: Microsoft Foundry and Azure AI Search, or Vertex AI (as assigned), cloud identity and secrets services, cost management tools

Week 6: Reliability, Production Readiness, Capstone & Panel Defence

Course Outcomes

- Instrument an AI system with tracing, logging and quality monitoring
- Define reliability objectives and fallback behaviour for an AI service
- Assess a system against reliability, governance and cost criteria
- Assemble an evidence pack: working system, evaluation results, governance mapping and cost report
- Defend a governed AI system on the chosen cloud to a panel against the rubric
- Capture next steps for applying the learning at work through the Transfer Card

Hands-on Labs

- Block A: Add tracing, monitoring and fallback behaviour
- Block A: Finalise the system and its evidence pack
- Block B: Capstone: present and defend the governed AI system to a panel

Masterclasses (Monday to Thursday)

- Reliability engineering for AI: observability, SLOs and fallbacks
- Production readiness: reliability, governance and cost evidence
- Capstone build clinic
- Defence preparation: presenting trade-offs to a panel

Tools / Demos: Learner's own capstone build, tracing and monitoring tools, evaluation framework, rubric and evidence pack template