AI Capability Programme: Full-Stack Engineering

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

License / Access Requirements

Paid

- **Azure Cloud subscription with LLM model access on the assigned cloud, with per-learner spend caps**
- **Approved AI assistant licences (Copilot Chat, ChatPwC); any additional coding assistant to be confirmed**

Free / open-source

- Node.js, Python, VS Code, Git, Docker (reference web stack to be confirmed)
- Open-source vector store, testing and evaluation libraries

Environment

- Per-learner sandbox on an approved stack, with budget alerts and a hard cost cap
- Stable internet connection and a second screen recommended for live coding

Course Objective

This OneCloud programme builds the capability to design, build, test and secure AI-native applications across the full stack. Participants gain hands-on expertise in AI-assisted and agentic development workflows, LLM integration, and retrieval-based features with citations and evaluation. They also become proficient with testing, code review, security and cost optimisation for AI features, and apply them to a verified, cost-capped AI feature built over the six weeks.

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

- Application engineers with three to six years' experience
- Full-stack and web developers who build applications without AI today
- Engineers moving from conventional to AI-native application development

Prerequisites

- Three to six years of application development experience
- Working knowledge of a web stack: front end, back end and REST APIs
- Familiarity with Git and version control
- Prior AI/ML experience is not required

Week-wise Course Content

Week 1: LLM Foundations, Prompting & AI-Assisted Development

Course Outcomes

- Explain how large language models work at the level an application engineer needs (tokens, context windows, sampling, limits)

- Apply prompting techniques (zero-shot, few-shot, system prompts) to produce reliable, structured responses
- Call an LLM API from a back-end service and handle streaming, errors and timeouts
- Estimate cost per interaction and latency from token usage and model choice
- Use an AI coding assistant to plan, generate, refactor and explain code in an existing application
- Write effective context for code tasks (specification, constraints, examples, acceptance checks)
- Recognise the limits of AI-generated code and apply a verification checklist before accepting it

Hands-on Labs

- Block A: Build a streaming AI chat feature with structured JSON output
- Block A: Measure tokens, latency and cost per request
- Block B: Add a feature to a sample web app using an AI coding assistant
- Block B: Refactor and document existing code with AI assistance

Masterclasses (Monday to Thursday)

- How LLMs work for application engineers
- Prompting and structured outputs
- AI-assisted development workflows
- Calling model APIs from the back end: streaming, errors and cost

Tools / Demos: Assigned-cloud LLM service, AI assistants (Copilot Chat, ChatPwC), web framework, Git, VS Code

Week 2: Full-Stack AI Integration, Embeddings & Retrieval Basics

Course Outcomes

- Design the architecture of an AI-enabled web application (front end, API layer, model gateway, data store)
- Implement conversation state, streaming and error handling across front end and back end
- Design user experiences for AI features: loading states, uncertainty, feedback and human fallback
- Manage prompts and model configuration as versioned, testable application assets
- Explain embeddings and vector similarity, and choose a chunking strategy for a document set
- Build a retrieval pipeline: ingestion, chunking, embedding, indexing and query
- Apply hybrid search and metadata filters to improve retrieval relevance

Hands-on Labs

- Block A: Build an AI assistant feature across front end and back end
- Block A: Add conversation memory and user feedback capture
- Block B: Ingest and chunk retail policy documents
- Block B: Create embeddings and index them in a vector store
- Block B: Implement hybrid search with metadata filters

Masterclasses (Monday to Thursday)

- Architecture of an AI-enabled web application
- Conversation state, UX patterns and feedback for AI features
- Embeddings, chunking and vector search
- Building a retrieval pipeline end to end

Tools / Demos: Assigned-cloud LLM service, embedding model, vector store, web framework, AI assistants, Git

Week 3: Retrieval-Based Features, Citations, Evaluation & Hackathon

Course Outcomes

- Design a retrieval-augmented generation (RAG) feature with grounded answers and a defined fallback when evidence is missing
- Implement source citations that link each claim in a response to a retrieved passage
- Build an evaluation set and measure retrieval quality and answer groundedness

- Compare chunking, retrieval and prompt variants using evaluation results rather than intuition
- Deliver a team build within a stated cost ceiling
- Record design decisions and each contributor's role in a team decision log
- Present and demonstrate a team build to reviewers

Hands-on Labs

- Block A: Build a RAG feature that returns cited answers
- Block A: Create a golden question set and score groundedness
- Block B: Hackathon: team build under a cost ceiling
- Block B: Demonstrate the build and submit the decision log

Masterclasses (Monday to Thursday)

- RAG feature design: grounding and answer generation
- Citations and source attribution in the user interface
- Evaluating retrieval-based features: golden sets, groundedness and relevance
- Hackathon briefing: brief, cost ceiling and team roles

Tools / Demos: Assigned-cloud LLM service, vector store, evaluation framework, web framework, cost tracking, Git

Week 4: Testing, Code Review & Security for AI-Native Applications

Course Outcomes

- Design a test strategy for AI features that combines unit, integration and evaluation-based tests
- Automate evaluation checks in a CI pipeline and set pass thresholds
- Review AI-generated code for correctness, maintainability, licensing and security issues
- Use AI-assisted code review and compare its findings with a human review
- Identify common LLM application risks (prompt injection, data leakage, insecure output handling, excessive agency)
- Apply input and output guardrails, PII redaction and secrets management
- Threat-model an AI feature and document its mitigations

Hands-on Labs

- Block A: Write unit and evaluation tests for the RAG feature
- Block A: Add an evaluation gate to a CI pipeline
- Block A: Review AI-generated pull requests against a checklist
- Block B: Run prompt injection tests against the feature
- Block B: Add guardrails and PII redaction
- Block B: Complete a threat model for the feature

Masterclasses (Monday to Thursday)

- Testing non-deterministic features
- Reviewing AI-generated code
- Security risks in LLM applications: prompt injection and data leakage
- Guardrails, content filtering and PII handling

Tools / Demos: CI pipeline, evaluation framework, test libraries, AI assistants, secrets manager, Git

Week 5: Agentic Development Workflows, Observability & Cost Optimisation

Course Outcomes

- Describe agentic development workflows and where human review checkpoints belong
- Direct a coding agent through a plan, implement, test and review loop using clear specifications
- Set boundaries for agent actions (permissions, scope, approvals) and review its changes
- Identify the main drivers of AI feature cost (tokens, context size, model choice, retries)
- Reduce cost per interaction using caching, model routing, prompt trimming and batching
- Add tracing, logging and usage metrics to an AI feature

- Set budgets and alerts that keep a feature within a cost cap

Hands-on Labs

- Block A: Write a specification and acceptance tests for a coding agent
- Block A: Deliver a feature using a coding agent with review checkpoints
- Block B: Add response caching and model routing
- Block B: Trace requests and measure cost per interaction
- Block B: Enforce a hard cost cap with alerts

Masterclasses (Monday to Thursday)

- Agentic software development workflows
- Supervising coding agents: context, boundaries and approvals
- Cost drivers and optimisation: caching, routing and batching
- Observability for AI features: tracing, logging and monitoring

Tools / Demos: AI assistants and coding agents (as approved), assigned-cloud LLM service, tracing and monitoring tools, cost dashboards, CI pipeline

Week 6: Production Readiness, Capstone Preparation & Panel Defence

Course Outcomes

- Assess an AI feature against a production-readiness checklist (quality, security, cost, observability)
- Assemble an evidence pack: working feature, evaluation results, security review and cost report
- Explain design decisions and trade-offs with reference to evidence
- Defend a verified, cost-capped AI feature to a panel against the rubric
- Capture next steps for applying the learning at work through the Transfer Card

Hands-on Labs

- Block A: Finalise the cost-capped AI feature and its evidence pack
- Block A: Run the final evaluation, security and cost checks
- Block B: Capstone: present and defend the feature to a panel

Masterclasses (Monday to Thursday)

- Production readiness for AI features
- The evidence pack and how the panel judges it
- Capstone build clinic
- Defence preparation: presenting trade-offs and evidence

Tools / Demos: Learner's own capstone build, evaluation framework, cost dashboards, rubric and evidence pack template