Agentic AI Engineering

A 3-Phase, 25-Module Learning Path

This curriculum takes a learner from Python fundamentals all the way to deploying production-grade, multi-agent AI systems. It is organised into three progressive phases: **Foundations**, where core tooling and LLM fundamentals are established; **Intermediate**, where orchestration frameworks, stateful workflows, and interoperability protocols are introduced; and **Advanced**, where reflection, multi-agent orchestration, safety guardrails, observability, fine-tuning, deployment, and a full capstone project bring everything together.

Phase	Focus	Modules
Phase 1	Foundations	6
Phase 2	Intermediate — Orchestration & Protocols	8
Phase 3	Advanced — Production, Safety & Capstone	11

Table of Contents

Phase 1: Foundations

1. Python & AI Development Environment Setup
2. Building AI Apps with FastAPI, Streamlit & Gradio
3. LLM Fundamentals, Context & Prompt Engineering
4. Embeddings, Vector Databases & Semantic Search
5. AI-Assisted (Vibe) Coding for Developers
6. No-Code Workflow Automation

Phase 2: Intermediate — Orchestration & Protocols

7. LangChain Core — Chains, Memory & RAG
8. LangChain Agents & Tool Use
9. LangGraph — Stateful Workflows & Routing
10. LangGraph — Cycles, Human-in-the-Loop & Persistence
11. Tracing, Evaluation & Testing for LLM Applications
12. Model Context Protocol — Architecture & Custom Servers
13. MCP — Ecosystem Integrations
14. Programmatic Prompting

Phase 3: Advanced — Production, Safety & Capstone

15. Deep Agents — Reflection, Planning & Long-Term Memory
16. Multi-Agent Orchestration & Custom MCP Servers
17. Agentic RAG & GraphRAG
18. Advanced Agentic Workflows with Visual Automation Tools
19. Guardrails & AI Safety
20. Monitoring, Evaluation & Fine-Tuning
21. Containerising & Deploying AI Agents
22. Capstone Project, Part 1 — Architecture, Build & Integration
23. Capstone Project, Part 2 — Deployment
24. Agent Interoperability Protocols
25. Generative AI and LLM Security

Phase 1
Foundations

Core Python, app-building tooling, LLM fundamentals, and retrieval basics — the building blocks every agentic AI engineer needs before working with orchestration frameworks.

Modules in this phase: 6

1. Python & AI Development Environment Setup

TOPICS COVERED

- Overview of the agentic AI ecosystem and where Python fits
- Setting up Python with pyenv for version management
- Creating and managing virtual environments using venv and conda
- Configuring VS Code with AI-friendly extensions and settings
- Setting up Jupyter Lab for interactive AI experimentation
- Understanding Python project structure for production AI apps
- Managing dependencies with pip and requirements.txt
- Securely managing API keys using python-dotenv and .env files
- Introduction to Git: initialising repos, branching, and committing
- Pushing AI projects to a Git host with a proper .gitignore
- Writing asynchronous Python with asyncio for concurrent LLM calls
- Comparing sequential vs concurrent API calls: performance benchmarking

HANDS-ON PRACTICE

- Setting up a Python AI project from scratch
- Managing API keys securely with python-dotenv
- Running concurrent LLM calls with asyncio

SKILLS COVERED

Python • Virtual Environments • Git & Version Control • asyncio • Project Structure • Secret Management

2. Building AI Apps with FastAPI, Streamlit & Gradio

TOPICS COVERED

- REST API fundamentals: HTTP methods, status codes, request/response lifecycle
- FastAPI project structure: routers, models, and dependency injection
- Defining Pydantic models for request validation and response schemas
- Building async FastAPI endpoints for LLM-powered routes
- Streaming LLM responses using Server-Sent Events (SSE)
- Handling file uploads in FastAPI for document processing pipelines
- Running and testing FastAPI with Uvicorn and Swagger UI
- Streamlit fundamentals: layout, widgets, session state, and caching
- Building a real-time streaming chat UI in Streamlit
- Gradio components: text, file, image inputs and markdown outputs

- Creating shareable Gradio demos and deploying them publicly
- Wiring a Streamlit or Gradio front-end to a FastAPI backend
- Comparing Streamlit vs Gradio: when to use which

HANDS-ON PRACTICE

- Building your first FastAPI LLM endpoint
- Streaming AI responses live in a Streamlit UI
- Creating a shareable AI demo with Gradio
- Connecting a Gradio front-end to a FastAPI backend

SKILLS COVERED

FastAPI • Streamlit • Gradio • REST API Design • Server-Sent Events • Uvicorn • Pydantic

3. LLM Fundamentals, Context & Prompt Engineering

TOPICS COVERED

- How large language models work: tokenisation, attention, and next-token prediction
- Understanding context windows: limits, costs, and implications for agents
- Chat completion APIs: parameters and response structure
- Comparing message formats, system prompts, and model differences across providers
- Structuring system prompts for domain-expert personas
- Zero-shot, one-shot, and few-shot prompting patterns
- Chain-of-Thought (CoT) prompting for complex reasoning tasks
- Tree-of-Thought (ToT) for branching multi-path reasoning
- ReAct prompting: combining reasoning and action steps
- Structured outputs using JSON mode and response schemas
- Function calling: defining tools and routing user intent
- Context management strategies: chunking, summarisation, and windowing
- Measuring and optimising token usage
- Evaluating and benchmarking LLM output quality

HANDS-ON PRACTICE

- Comparing zero-shot, few-shot, and chain-of-thought prompts
- Extracting structured data with JSON mode and Pydantic
- Routing user intent with function calling

SKILLS COVERED

Prompt Engineering • LLM APIs • Context Management • Function Calling • JSON Mode • Token Optimisation

4. Embeddings, Vector Databases & Semantic Search

TOPICS COVERED

- What vector embeddings are and why they matter for AI agents
- Embedding models: proprietary APIs and open-source sentence-transformers
- Generating embeddings for text, documents, and structured data
- Vector database architecture: collections, documents, metadata, and IDs
- CRUD operations and metadata filtering in a vector store
- Vector index types: Flat, IVF, HNSW — trade-offs explained
- Building and querying a vector index from a document corpus

- Nearest-neighbour retrieval: understanding similarity metrics (cosine, L2, dot product)
- Hybrid search: combining keyword scoring with semantic vector scores
- Chunking strategies for effective embedding: fixed, sentence, and recursive
- Benchmarking vector stores: speed, accuracy, and scalability
- Visualising embedding clusters to understand semantic groupings
- RAG architecture overview: retriever, generator, and the full pipeline

HANDS-ON PRACTICE

- Building a semantic search engine with a vector database
- Benchmarking flat vs HNSW retrieval speed
- Implementing hybrid keyword and semantic search
- Visualising document embeddings

SKILLS COVERED

Vector Embeddings • Vector Databases • Semantic Search • Hybrid Search • sentence-transformers • RAG Fundamentals

5. AI-Assisted (Vibe) Coding for Developers

TOPICS COVERED

- Vibe coding fundamentals and mindset
- Overview of AI coding assistant tools
- Setting up and configuring an in-editor AI coding assistant
- Chat-based and interactive coding workflows
- Code quality and testing with AI assistance
- Advanced techniques for AI-assisted coding
- Codebase-aware AI editors: fundamentals
- Understanding codebase intelligence and context retrieval
- Multi-file composition and refactoring with an AI editor
- End-to-end feature development with an AI-assisted workflow

HANDS-ON PRACTICE

- Working with an AI coding assistant in your editor
- Building a small project with a codebase-aware AI editor

SKILLS COVERED

AI-Assisted Coding • In-Editor AI Assistants • Codebase-Aware Editors

6. No-Code Workflow Automation

TOPICS COVERED

- No-code automation platform fundamentals and workflows
- Trigger and action patterns
- Multi-step workflows for complex logic
- Using automation-platform APIs for custom integrations
- Built-in tables for lightweight agent data storage
- A second automation platform: overview and modules
- Advanced scenario routing and branching logic
- Webhooks and API integration within no-code tools
- Connecting Python agents to no-code automation platforms

- Scaling no-code automation for production use

HANDS-ON PRACTICE

- Working with a no-code automation platform end to end

SKILLS COVERED

No-Code Automation • Workflow Triggers & Actions • No-Code Integrations • Hybrid Agent-Automation

Phase 2

Intermediate — Orchestration & Protocols

Chains, memory, agents, stateful graph workflows, evaluation tooling, and the Model Context Protocol — where single-step LLM calls become real orchestrated agent systems.

Modules in this phase: 8

7. LangChain Core — Chains, Memory & RAG

TOPICS COVERED

- LangChain architecture: runnables, the pipe operator, and the LCEL paradigm
- Building chains with LCEL: prompt | llm | parser patterns
- Prompt templates: chat prompt templates, message placeholders, and partial templates
- Output parsers: string, JSON, and Pydantic-based parsers
- Conversational memory types: buffer memory, summary memory, vector-store memory
- Maintaining multi-turn conversation history across sessions
- Document loaders: PDF, web, CSV, and custom loaders
- Text splitters: recursive character splitting and semantic chunking
- Building a RAG pipeline: loader, splitter, embedder, retriever, generator
- Retriever types: vector-store retriever, multi-query retriever, contextual compression retriever
- Callbacks and hooks for logging and monitoring
- Chaining multiple retrievers and combining results

HANDS-ON PRACTICE

- Building an LCEL chain with memory for multi-turn conversations
- Creating a RAG pipeline over a PDF knowledge base
- Comparing multi-query vs single-query retrieval quality

SKILLS COVERED

LangChain • LCEL • RAG Pipelines • Conversational Memory • Document Loading • Text Splitting • Retrieval Strategies

8. LangChain Agents & Tool Use

TOPICS COVERED

- Agent fundamentals: perception, reasoning, action, and observation loop
- ReAct agent pattern: interleaving reasoning traces and tool calls
- Plan-and-Execute pattern: generating a plan first, then executing steps
- Built-in agent tools: web search, code execution, and knowledge lookups
- Creating custom tools with decorators and structured tool classes
- Defining tool input schemas with Pydantic for safe structured arguments
- Agent executor configuration: max iterations, early stopping, and verbose mode
- Structured output agents: forcing final answers into typed Pydantic objects
- Connecting agents to SQL databases for natural-language queries
- Multi-step reasoning: chaining tool outputs as inputs to subsequent tool calls

- Handling tool failures: try/except patterns and automatic retry logic
- Streaming intermediate agent steps to a UI in real time

HANDS-ON PRACTICE

- Building a ReAct agent with web search and code-execution tools
- Connecting an agent to a SQL database for natural language queries
- Adding retry logic and error handling to a tool-calling agent

SKILLS COVERED

LangChain Agents • ReAct Pattern • Custom Tool Development • Agent Executor • SQL Agent • Structured Output • Tool Error Handling

9. LangGraph — Stateful Workflows & Routing

TOPICS COVERED

- Why LangGraph: limitations of linear chains for complex agentic workflows
- Core LangGraph concepts: nodes, edges, and the state schema
- Defining a typed state and passing it through the graph
- Building a state graph: adding nodes, setting entry points, and compiling
- Conditional edges: routing to different nodes based on state values
- Message-graph pattern for chat-style workflows with message history
- Streaming node-by-node output during graph execution
- Adding human-readable labels and descriptions to graph nodes
- Visualising and debugging graphs with a graph studio
- Building a specialist routing system: classifier node + expert nodes
- Handling terminal states: defining end conditions and exit paths
- Composing reusable node functions with clean state interfaces
- Testing individual nodes in isolation before wiring the full graph

HANDS-ON PRACTICE

- Building a conditional routing graph with specialist nodes
- Streaming live node-by-node output
- Debugging and visualising a graph in a graph studio
- Building an order-processing state machine with fail and retry states

SKILLS COVERED

LangGraph • StateGraph • Conditional Routing • Graph Compilation • Graph Debugging Tools • Stateful Agent Design • Node Streaming

10. LangGraph — Cycles, Human-in-the-Loop & Persistence

TOPICS COVERED

- Cycles and loops: when and why to add feedback loops to a graph
- Implementing self-correction loops: generate, evaluate, revise
- Human-in-the-loop (HIL): understanding interrupt-before and interrupt-after patterns
- Pausing a graph, collecting human input, and resuming from a checkpoint
- Building approval-gate workflows: draft, review, approve, publish
- Checkpointers: what they are and why persistence matters for agents
- Lightweight checkpointer setup, configuration, and state recovery
- High-throughput checkpointer setup for production environments

- Resuming an interrupted workflow after a simulated crash
- Fan-out APIs: dispatching work to parallel worker nodes
- Aggregating parallel node results back into a shared state
- Sub-graphs: building modular graph components and composing them
- Sharing state across parent and sub-graphs safely

HANDS-ON PRACTICE

- Adding a human approval gate with interrupt and resume
- Implementing checkpointing and crash recovery
- Fanning out tasks to parallel nodes

SKILLS COVERED

Human-in-the-Loop • LangGraph Checkpointers • State Persistence • Parallel Execution • Sub-graph Composition • Cycle Design

11. Tracing, Evaluation & Testing for LLM Applications

TOPICS COVERED

- Why observability matters for LLM applications in production
- Tracing platform overview: projects, runs, traces, and feedback
- Instrumenting LangChain and LangGraph runs with tracing
- Inspecting individual spans: inputs, outputs, latency, and token cost
- Creating evaluation datasets: golden QA pairs and edge cases
- Running automated evaluations with built-in evaluators
- LLM-as-judge evaluation: writing a custom grading prompt
- Heuristic evaluators: rule-based checks for format, length, and keywords
- Regression testing: comparing two prompt versions on the same dataset
- A/B testing agent behaviours: setting up experiments and tracking metrics
- Annotating runs with human feedback for future dataset creation
- Production monitoring dashboards: latency, error rate, and cost over time
- Setting up alerts for anomalous agent behaviour in production

HANDS-ON PRACTICE

- Tracing a LangGraph agent end-to-end
- Running an automated evaluation suite with LLM-as-judge
- Setting up an A/B experiment to compare two agent strategies

SKILLS COVERED

LLM Tracing • Automated Evaluation • LLM-as-Judge • Regression Testing • A/B Testing • Production Monitoring

12. Model Context Protocol — Architecture & Custom Servers

TOPICS COVERED

- What the Model Context Protocol is and why it was created
- MCP architecture: hosts, clients, and servers — roles and responsibilities
- The three MCP primitives: resources, tools, and prompts
- The JSON-RPC 2.0 transport layer: stdio and HTTP/SSE communication
- Setting up an MCP SDK and scaffolding a minimal server
- Defining and exposing custom tools from an MCP server
- Defining resource endpoints that expose live or static data sources

- Implementing prompt templates as MCP primitives
- Tool discovery and capability negotiation between client and server
- Connecting a custom MCP server to a desktop AI assistant
- Connecting a custom MCP server to an IDE-based assistant
- MCP security model: permission scopes and access control
- Server-side input validation and request sanitisation
- Debugging MCP servers with logging and inspector tooling

HANDS-ON PRACTICE

- Scaffolding and connecting your first custom MCP server
- Exposing a live data API as an MCP resource endpoint
- Implementing permission checks and input validation
- Debugging an MCP server with inspector tooling

SKILLS COVERED

Model Context Protocol • MCP SDK • Custom MCP Servers • Desktop Assistant Integration • Tool Discovery • MCP Security • JSON-RPC

13. MCP — Ecosystem Integrations

TOPICS COVERED

- Overview of the MCP ecosystem: official and community servers
- Database MCP servers: exposing schema, running queries, and returning results
- Filesystem MCP servers: safe file read, write, list, and move operations
- Source-control MCP servers: managing issues, pull requests, branches, and comments
- Browser-automation MCP servers: automation and structured data extraction
- Chaining multiple MCP servers in a single assistant session
- Using MCP servers as tools inside LangChain agents
- Using MCP servers as tools inside LangGraph nodes
- Integrating MCP into a multi-agent framework as an external tool
- Building a report pipeline: database MCP query to filesystem MCP write
- Security considerations when chaining multiple MCP servers
- Production deployment patterns for MCP servers

HANDS-ON PRACTICE

- Chaining a database MCP server and a filesystem MCP server to build a report pipeline
- Managing a source-control repository via an MCP server
- Automating browser data extraction with an MCP server

SKILLS COVERED

Database MCP • Source-Control MCP • Browser-Automation MCP • Filesystem MCP • Multi-MCP Chaining • MCP + LangGraph • MCP + Multi-Agent Frameworks

14. Programmatic Prompting

TOPICS COVERED

- Introduction to programmatic prompting frameworks
- Modules and signatures for declarative prompt programs
- Chain-of-thought and predict-style modules
- ReAct-style modules within a programmatic framework

- Automated prompt optimisers
- Compiling and validating programmatic prompt programs
- Writing metric functions and evaluation logic
- Working across multiple LLM providers
- Building self-improving prompt pipelines
- Programmatic prompting vs manual prompting: trade-offs
- Deploying programmatic-prompting systems to production

HANDS-ON PRACTICE

- Building a self-improving prompt optimisation system

SKILLS COVERED

Programmatic Prompting • Prompt Optimisation • Self-Improving Systems

Phase 3

Advanced — Production, Safety & Capstone

Deep agents, multi-agent orchestration, agentic and graph-based RAG, guardrails, observability, fine-tuning, containerised deployment, interoperability, security, and a full capstone build.

Modules in this phase: 11

15. Deep Agents — Reflection, Planning & Long-Term Memory

TOPICS COVERED

- What makes an agent 'deep': meta-cognition and self-awareness in LLMs
- Self-reflection loops: generate, critique against a rubric, revise
- Multi-turn reflection: iterating until a quality threshold is met
- Plan-and-Execute pattern: separating planning from execution
- Replanning: updating the plan mid-execution when new information arrives
- ReAct vs Plan-and-Execute: trade-offs and when to use each
- Long-term memory types: episodic, semantic, and procedural
- Building an episodic memory store with a stateful graph and a vector database
- Memory consolidation: summarising and compressing old episodic memories
- Semantic memory: storing facts and updating them across sessions
- Cross-session memory retrieval: using similarity search to recall past context
- Confidence and uncertainty estimation: flagging low-confidence answers
- Agent self-improvement: using evaluation scores to update future behaviour
- Combining reflection, planning, and memory into one unified agent architecture

HANDS-ON PRACTICE

- Building a self-critique and revision loop
- Implementing a Plan-and-Execute agent with mid-run replanning
- Adding cross-session episodic memory to an agent

SKILLS COVERED

Self-Reflection Loops • Plan-and-Execute • Episodic Memory • Semantic Memory • Memory Consolidation • Confidence Estimation • Deep Agent Architecture

16. Multi-Agent Orchestration & Custom MCP Servers

TOPICS COVERED

- Why build custom MCP servers for orchestration frameworks
- Lightweight MCP framework overview
- Defining MCP tools, resources, and prompts
- Type validation with Pydantic
- Error handling in MCP servers
- MCP server deployment patterns
- Testing MCP servers

- Multi-agent orchestration: agent roles, crews, and delegation flow
- Building a custom MCP server from scratch and wiring it into a multi-agent crew

HANDS-ON PRACTICE

- Building a custom MCP server from scratch

SKILLS COVERED

Multi-Agent Orchestration • Lightweight MCP Frameworks • Custom MCP Servers • Pydantic Validation • Server Testing

17. Agentic RAG & GraphRAG

TOPICS COVERED

- RAG failure modes: hallucination, missed retrieval, and irrelevant context
- Diagnosing RAG failures systematically with retrieval-evaluation metrics
- Agentic RAG: the agent decides whether, what, and how to retrieve
- Self-RAG: retrieve, score chunk relevance, decide to regenerate, synthesise
- Corrective RAG: falling back to web search when the local knowledge base fails
- Advanced chunking strategies: semantic chunking, late chunking, parent-child chunking
- Re-ranking retrieved chunks with cross-encoder models
- Hypothetical document embeddings: generating a hypothetical answer to improve query embedding
- Query expansion: generating multiple reformulations to broaden recall
- GraphRAG: representing documents as a knowledge graph of entities and relations
- Building a knowledge graph from documents
- Multi-hop reasoning: traversing the graph to answer complex questions
- Hybrid RAG: combining graph traversal with vector similarity search
- Evaluating RAG pipelines: faithfulness, answer relevancy, context recall
- Benchmarking agentic RAG vs standard RAG on the same test set

HANDS-ON PRACTICE

- Implementing Self-RAG with relevance scoring and conditional regeneration
- Building a corrective RAG pipeline with web-search fallback
- Constructing a knowledge graph and running multi-hop GraphRAG queries
- Evaluating a RAG pipeline end-to-end

SKILLS COVERED

Agentic RAG • Self-RAG • Corrective RAG • GraphRAG • Knowledge Graphs • Re-ranking • HyDE • RAG Evaluation

18. Advanced Agentic Workflows with Visual Automation Tools

TOPICS COVERED

- Visual workflow automation overview and its role in agentic systems
- Self-hosting a workflow engine with Docker and configuring for production
- Core concepts: workflows, nodes, triggers, and credentials
- Webhook trigger nodes: receiving external events to kick off workflows
- LLM provider nodes: prompt construction, model selection, and response parsing
- HTTP request nodes: calling any external API from a workflow
- Notification nodes: sending formatted messages and alerts
- Error handling: error branches, retry logic, and fallback paths
- Scheduled triggers: running agent workflows on a timetable

- Workflow expressions: transforming data between nodes
- Integrating a visual workflow engine with LangGraph via webhook nodes
- Storing workflow outputs in external spreadsheets and databases
- Deploying workflows to production and managing credentials securely

HANDS-ON PRACTICE

- Building a webhook-triggered lead-enrichment workflow with an LLM node
- Adding error handling and retry logic to a workflow
- Scheduling a daily monitoring workflow with a scheduled trigger and an LLM

SKILLS COVERED

Visual Workflow Automation • Webhook Triggers • LLM Node Integration • Error Handling • Scheduled Automation • Workflow + LangGraph Integration

19. Guardrails & AI Safety

TOPICS COVERED

- The AI safety landscape: why guardrails are non-negotiable in production
- Understanding the most critical LLM application risks
- Prompt injection attacks: anatomy, real examples, and detection strategies
- Rail-based guardrail frameworks: architecture and dialog-flow languages
- Writing rails that define allowed and blocked topics
- Input rails: validating user messages before the LLM sees them
- Output rails: filtering and validating LLM responses before delivery
- Validator-based guardrail frameworks: validators and the guard object pattern
- Built-in validators: toxic language, secrets, competitor mentions
- Writing custom validators for domain-specific rules
- PII detection: entities, recognisers, and anonymisation
- Building an end-to-end safety stack: input rail, LLM, output validator, PII scrub
- Toxicity, bias, and hallucination detection tools and integrations
- Constitutional AI principles: embedding safety constraints in agent design
- Testing the safety stack: red-teaming with adversarial prompt suites

HANDS-ON PRACTICE

- Writing rails to block off-topic and harmful requests
- Adding PII detection and redaction to an agent pipeline
- Red-teaming an agent with prompt injections and hardening its defences

SKILLS COVERED

Guardrail Frameworks • Dialog-Flow Rails • Validator-Based Guardrails • PII Detection • Prompt Injection Defence • AI Safety Design • LLM Risk Categories

20. Monitoring, Evaluation & Fine-Tuning

TOPICS COVERED

- The three pillars of LLM observability: logs, metrics, and distributed traces
- Instrumenting LangGraph agents with spans and attributes
- Exporting traces and interpreting the trace waterfall
- Metric collection: scrape targets, metric types, and query basics
- Building dashboards for latency, token cost, and error rate

- LLM-specific monitoring: embedding drift and data quality
- Cost tracking per run: token budgets, alerts, and quota management
- SLA and SLO design for agent systems: what to measure and why
- Fine-tuning fundamentals: when prompting is insufficient and fine-tuning is warranted
- Low-Rank Adaptation (LoRA): intuition, hyperparameters, and implementation
- Quantised fine-tuning for resource-constrained environments
- Building a fine-tuning dataset from agent traces
- Training a LoRA adapter on a custom domain dataset
- Evaluating a fine-tuned model vs the base model on a held-out test set
- Reinforcement learning from human feedback and preference optimisation: overview and concepts

HANDS-ON PRACTICE

- Instrumenting a LangGraph agent and building a monitoring dashboard
- Detecting embedding drift in a RAG pipeline
- Fine-tuning a model with LoRA and benchmarking against the base model

SKILLS COVERED

Distributed Tracing • Metrics & Dashboards • LLM Observability • LoRA • Quantised Fine-Tuning • Parameter-Efficient Fine-Tuning • SLA/SLO Design

21. Containerising & Deploying AI Agents

TOPICS COVERED

- Container fundamentals: images, containers, layers, and the build cache
- Writing a production container file for a Python FastAPI + LangChain application
- Multi-stage builds: separating build-time dependencies from the runtime image
- Multi-service orchestration: defining service, network, and volume configs
- Composing a full agent stack: API, vector database, cache, and a worker service
- Environment variable management: env files, secrets, and runtime injection
- Container health checks: liveness, readiness, and graceful shutdown patterns
- Pushing images to a container registry
- Deploying a containerised agent to a serverless container platform with auto-scaling
- Deploying to a managed container-orchestration service
- CI/CD pipelines: build, test, push, and deploy for AI services
- Blue/green deployments: switching traffic between versions with zero downtime
- Canary deployments: gradually rolling out a new agent version
- Monitoring deployed containers: logs, metrics, and alerting

HANDS-ON PRACTICE

- Building and running a multi-service agent stack
- Writing a CI/CD pipeline for an AI agent service
- Deploying a containerised agent with zero downtime
- Implementing a blue/green deployment switch

SKILLS COVERED

Containerisation • Multi-Service Orchestration • CI/CD • Serverless Container Deployment • Managed Container Orchestration • Blue/Green Deployment • Container Security

22. Capstone Project, Part 1 — Architecture, Build & Integration

TOPICS COVERED

- Capstone problem selection: choosing a real-world industry use case to solve
- System architecture design: identifying components, data flows, and boundaries
- Technology selection: reasoning through single-agent vs multi-agent vs hybrid approaches
- Designing the RAG layer: index strategy, chunking method, and retrieval approach
- Designing the multi-agent layer: agent roles, task decomposition, and delegation flow
- Integrating MCP servers: identifying which external tools the system needs
- Wiring guardrails into the architecture: input rails, output validators, PII scrubbing
- Building the core agent loop: RAG + stateful graph + tool use running end-to-end
- Connecting the front-end: a UI wired to the backend agent service
- Instrumenting tracing across all components
- Adding monitoring dashboards: latency, token cost, and error-rate views
- Peer design review: presenting the architecture and receiving structured feedback
- Iterating on the build based on review feedback before integration testing

HANDS-ON PRACTICE

- Architecting the full capstone system and presenting the design
- Building the core agent loop with RAG, multi-agent orchestration, and guardrails
- Wiring tracing and monitoring into the full system
- Running end-to-end integration tests across all components

SKILLS COVERED

System Architecture Design • RAG Layer Design • Multi-Agent Design • MCP Integration • Guardrails Integration • LLM Tracing • Monitoring Dashboards

23. Capstone Project, Part 2 — Deployment

TOPICS COVERED

- Containerising all capstone services: writing production container files for each component
- Writing the multi-service stack: linking the API, vector database, cache, and agent worker
- CI/CD pipeline for the capstone: automated testing, build, and cloud deployment
- Deploying the full stack to a cloud environment
- Performance testing: measuring latency, throughput, and cost per query under load
- Security review: checking for prompt injection vulnerabilities, data leakage, and access control gaps
- Writing technical documentation: README, API spec, and architecture decision records
- Writing a project brief: summarising the problem, approach, results, and trade-offs
- Peer code review: reviewing a peer's codebase and providing structured written feedback
- Incorporating peer review feedback and shipping the final version
- Preparing a live demo narrative: telling the story of the system to a non-technical audience
- Live demo delivery: presenting the working system with a monitoring dashboard live
- Capstone retrospective: what worked, what you would do differently, and key takeaways

HANDS-ON PRACTICE

- Deploying the full capstone stack to the cloud with CI/CD
- Running a performance and security audit on the live system
- Delivering the live system demo with real-time monitoring

SKILLS COVERED

Multi-Service Deployment • CI/CD Deployment • Cloud Deployment • Performance Testing • Security Review • Technical Documentation • Live Demo Delivery

24. Agent Interoperability Protocols

TOPICS COVERED

- The agent interoperability challenge
- Agent-to-agent communication protocol overview
- SDKs and reference implementations for agent-to-agent communication
- Agent communication protocols for tool-using agent frameworks
- Agent network protocols using decentralised identifiers
- Agent cards for capability discovery
- Cross-framework agent communication
- Building multi-protocol gateways
- Security and trust in agent networks
- Enterprise agent federation
- Emerging standards and governance for agent interoperability

HANDS-ON PRACTICE

- Building an interoperable multi-framework agent network

SKILLS COVERED

Agent Protocols • Cross-Framework Interop • Agent Networks • Protocol Governance

25. Generative AI and LLM Security

TOPICS COVERED

- Threats in generative AI systems
- Common attack vectors in generative AI systems
- Model theft and extraction attacks
- Mitigation strategies for generative-AI risks
- LLM-specific threats and risks
- Aligning LLM output to security objectives
- Securing AI training data and pipelines
- Risks in AI model hubs and repositories
- Dependency scanning and third-party model risks
- Bias, fairness, and ethical design in AI systems
- Regulatory and compliance standards for AI systems
- Multimodal AI threat intelligence
- Defending cyber operations with agentic AI

HANDS-ON PRACTICE

- Detecting prompt injection and jailbreak risks
- Integrating a large language model API into a security-review workflow
- Securing AI data against poisoning risks
- Tracking model provenance and scanning dependencies
- Ethical screening of AI system outputs
- Using agentic AI for cybersecurity triage

SKILLS COVERED

GenAI Security & Threat Analysis • LLM Risk Assessment & Mitigation • AI Pipeline & Data Security • Ethical AI & Compliance