

COURSE OUTLINE

Forward-Deployed AI Engineering (FDE-2) — Design, Build, and Deliver Enterprise AI Solutions

An intensive, lab-driven program for engineers who design, integrate, productionize, and deliver AI solutions inside real enterprise environments

DURATION

3 days

FORMAT

Instructor-led, lab-driven, capstone project

DELIVERY

Online Instructor Led Workshop · Onsite Program for your Teams

TRAINER

Muhammed Shakir

Why this course exists

Knowing how to call an LLM API is not the same as being able to walk into an enterprise, understand its business problem, and deliver an AI solution that stakeholders trust, security approves, and operations can run. Forward-deployed engineers sit in that gap: they gather requirements, choose the right architecture, build agentic systems that touch real business systems, and carry the work through evaluation, security review, and go-live.

This program turns AI engineering into a client-ready delivery discipline: participants design, build, integrate, and operate a multi-agent enterprise AI solution end to end, then deliver it the way a forward-deployed engineer would deliver it to a client.

By the end of three days, participants will have run a solution-design workshop, built a retrieval-grounded and agentic assistant wired into real enterprise-style APIs and databases, added the evaluation, observability, guardrail, and security controls a production system needs, and presented the finished solution as a client-facing deployment pack.

What makes this different

Every technical decision here is tied to a business problem, an integration point, or a delivery risk. **This is a delivery room, not a framework tour: each capability is built against a real client scenario and reviewed the way a client would review it.** You will:

- **Design before you build** — run a discovery workshop, translate business pain into an opportunity statement with KPIs and acceptance criteria, and choose the smallest AI solution pattern that actually solves it.
- **Build agentic systems that touch the business, not just documents** — go from a single retrieval-augmented assistant to a bounded agent to a multi-agent system orchestrated with LangChain and LangGraph, calling real enterprise APIs and databases.
- **Treat production concerns as part of the deliverable** — evaluation, observability, guardrails, security, and scalability are built alongside the feature, not added at the end.
- **Close the loop with the client** — every increment ends in something you could show a stakeholder: a demo, an evaluation report, a risk note, or a rollout plan.

The course favours one coherent enterprise deliverable over a scattered tour of tools.

Skills You'll Walk Away With

- Run a discovery workshop, translate ambiguous business pain into an AI opportunity statement, and select between automation, RAG, single-agent, and multi-agent solution patterns using value, feasibility, and risk.
- Design an enterprise AI solution architecture that names its data sources, integration points, trust boundaries, and success measures before a line of retrieval or agent code is written.
- Build and evaluate a retrieval-augmented assistant grounded in enterprise documents, structured databases, and permission-aware access.
- Build bounded AI agents and orchestrate them into a multi-agent system with LangChain and LangGraph, including tool use, memory, handoffs, approval checkpoints, and escalation exits.
- Integrate an AI solution with enterprise-style REST APIs, relational and document databases, and cloud deployment targets.
- Apply production discipline: evaluation harnesses, observability and tracing, guardrails against prompt injection and data leakage, authorization enforcement, and scalability and cost controls.
- Package and present an AI solution as a client-facing deliverable: architecture note, evaluation evidence, risk register, rollout plan, and stakeholder demo.

Who should attend

This program is for technically minded professionals who want to design, build, and deliver AI solutions inside customer or business environments, not just prototype them.

- **Forward-deployed and field engineers** — building and integrating AI systems directly inside customer environments.
- **AI and backend developers** — moving from single-model integrations to agentic, multi-system enterprise solutions.
-

Solution architects and technical consultants — choosing architectures, integration patterns, and defending delivery trade-offs to clients.

- **Pre-sales and solutions engineers** — translating client conversations into technically credible AI solution designs and demos.
- **Delivery leads and technical project managers** — overseeing AI projects that must reach production, not just a demo.

Prerequisites: comfort with programming, HTTP APIs, Git, basic SQL, and calling an LLM API. Prior exposure to RAG or agent frameworks is helpful but not required; LangChain, LangGraph, and orchestration concepts are introduced in plain engineering language.

Course Outline

Day 1 — Discover the Problem and Design the Enterprise AI Solution

Translate a real business problem into a testable AI opportunity, choose the right solution pattern, and ground the solution in enterprise data.

Session 1 — From Business Problem to AI Solution Design

Run the discovery workshop, define success, and design the architecture before writing retrieval or agent code.

- The forward-deployed engineer's operating model: discovery, solution design, build, evaluate, secure, deploy, and drive adoption as one accountable lifecycle
- Requirements gathering and workshop technique: discovery questioning, workflow mapping, identifying decision-makers, and surfacing unstated constraints
- Translating business pain into an AI opportunity statement: KPIs, acceptance criteria, value hypothesis, and a testable success measure
- Solution pattern selection: automation vs. LLM application vs. RAG vs. single agent vs. multi-agent, weighed on value, feasibility, effort, and risk
- Enterprise AI solution architecture: naming the data sources, APIs, databases, identity, and trust boundaries the solution will touch, before build starts
- Threat modeling early: prompt injection, data leakage, unauthorized tool use, and access-control gaps across ingestion, retrieval, agent action, and logging
- *Hands-on:* Run the capstone's discovery workshop and lock its architectural baseline.
 - Role-play a discovery workshop for **NorthBridge Industrial**, a fictional manufacturing company whose HR and IT service desks are overloaded with policy questions, status lookups, and routine requests. Capture pain points, stakeholders, and constraints in `docs/discovery-notes.md`.
 - Write the AI opportunity statement in `docs/opportunity-statement.md`: the target workflow, KPIs (deflection rate, time-to-resolution, escalation accuracy), and acceptance criteria.
 -

Create the project skeleton in `/Users/shakir/learning-capstones/forward-deployed-engineering-fde-2`, including environment configuration, secret-loading boundaries, a runnable local development command, and a health endpoint.

- Design the architecture note in `docs/architecture.md`: request 'agent orchestration' retrieval index 'HRIS/ticketing APIs' database 'logging, naming every trust boundary and where prompt injection, data leakage, and unauthorized action can enter.
- Draft the evaluation question set in `evaluation/v1.jsonl`: answerable policy questions, exact-record lookups, action requests (submit leave, raise a ticket), ambiguous requests, and requests that must escalate to a human. Add expected source or system references where known.
- **Session checkpoint:** By the end of this session, you have discovery notes, an opportunity statement with KPIs, an architecture note naming every integration point and trust boundary, a running skeleton application with a health endpoint, and a first evaluation set. No retrieval or agent behaviour is implemented yet.

Session 2 — Ground the Solution in Enterprise Data

Build a searchable, permission-aware knowledge base and connect it to the enterprise systems the solution must integrate with.

- Document ingestion and chunking for policy handbooks; preserving headings, section references, and source provenance
- Embeddings and vector search; a single MongoDB deployment holding application data, chunks, embeddings, and later agent traces, with Pinecone, Weaviate, and pgvector retained as conceptual comparisons
- Structured data integration: connecting to a relational employee/ticket database and to REST-style HRIS and ticketing APIs that stand in for real enterprise systems
- Metadata, permissions, and role-aware retrieval: HR content vs. IT content, employee-level vs. manager-level records, enforced before evidence reaches the model
- Semantic-search baseline: similarity scores, top-k retrieval, metadata filters, and inspecting irrelevant or missing results
- Choosing between document retrieval, direct API/database calls, and clarification or escalation for a given request type
- *Hands-on:* Build the reproducible knowledge base and connect the enterprise systems the capstone will act on.
 - Parse `raw/HR_Policy_Handbook.pdf` and `raw/IT_Support_Runbook.pdf` into `data/policy_sections.jsonl` with source, section, title, text, and source-hash metadata.
 - Stand up the mock enterprise systems: a FastAPI service backed by SQLite simulating the HRIS (employee records, leave balances) and the ticketing system (open tickets, status), documented in `docs/enterprise-systems.md`.

- Create idempotent MongoDB collection setup, validation rules, and vector-search indexes for policy chunks and embeddings, keeping department, role, and access attributes as filterable metadata.
- Implement semantic retrieval and a source-returning API endpoint; retrieve chunks only after applying the caller's role and department filters.
- Implement validated, read-only structured lookups against the mock HRIS and ticketing APIs for exact record requests (leave balance, ticket status).
- Write ingestion and retrieval tests, including two synthetic users (an employee and a manager) whose accessible records and policy sections differ.
- **Session checkpoint:** a repeatable ingestion pipeline, a working semantic-RAG endpoint over policy content, and validated structured lookups against the mock HRIS and ticketing APIs — all permission-aware and independently testable.

Day 2 — Build Agentic and Multi-Agent Enterprise Workflows

Move from a single grounded assistant to a bounded agent, then to a multi-agent system that plans, calls enterprise tools, and knows when to escalate.

Session 3 — Agents, Tools, and Multi-Agent Orchestration

Build a single tool-using agent, then decompose it into a coordinated multi-agent system with LangChain and LangGraph.

- From LLM application to agent: planning, memory, tool/function calling, and the retrieve 'act' inspect 'refine' loop
- Bounded agent design: approved tools only, validated inputs, permission checks, step limits, timeouts, and clarification or escalation exits
- LangChain fundamentals: tool definitions, structured output, and chains as the building blocks of an agent
- Multi-agent orchestration with LangGraph: role specialization, shared state, conditional routing, hand-offs between agents, and human-approval checkpoints before consequential actions
- Enterprise tool integration patterns: authenticated calls to the HRIS and ticketing APIs, idempotent writes, retries, and tool failure contracts independent of model instructions
- Pattern comparison: single retrieval assistant vs. single agent vs. multi-agent orchestration, weighed against task complexity, latency, and coordination cost
- *Hands-on:* Build the NorthBridge service-desk assistant as a bounded, then multi-agent, system over the Day 1–2 foundation.
 - **Single agent:** build one bounded agent with tools for semantic retrieval, structured lookup, and read-only status checks. Enforce step limits, timeouts, and a clarification exit; test it against ambiguous and out-of-scope requests.
 - **Triage agent:** classify an incoming request as policy question, record lookup, action request, or escalation, and route accordingly. Log the routing decision and its reason.

- **Retrieval agent:** wrap the Session 2 semantic-RAG and structured-lookup endpoints as a tool-using specialist that returns cited evidence.
- **Action agent:** implement validated, approval-gated write actions against the mock HRIS/ticketing APIs (submit a leave request, raise a ticket), requiring explicit human approval before any write commits.
- **Orchestration:** wire triage, retrieval, and action agents into a LangGraph graph with conditional edges, shared state, an escalation exit to a human, and full step-by-step tracing.
- Extend the evaluation harness to run the same question set against the single-agent and multi-agent configurations; compare task success, unnecessary escalations, and latency.
- **Session checkpoint:** a working multi-agent NorthBridge assistant — triage, retrieval, and action agents orchestrated with LangGraph, tool calls traced end to end, approval-gated writes, and a comparison against the single-agent baseline.

Day 3 — Productionize, Secure, and Deliver the Solution

Add the evaluation, observability, guardrail, security, and delivery discipline that turns a working demo into a client-ready AI solution.

Session 4 — Make It Production-Grade: Evaluation, Observability, Guardrails, and Security

Prove the solution is reliable, observable, safe, and secure enough to hand to a client's operations team.

- Evaluation suites and regression checks: retrieval relevance, answer faithfulness, citation support, task success for agent actions, latency, token use, and cost
- Observability: tracing every routing decision, retrieved passage, tool call, agent step, and outcome, with sensitive-content redaction
- Guardrails: input/output validation, PII handling, prompt-injection and jailbreak resistance, and tool-permission boundaries enforced independently of model instructions
- Security controls: authorization enforcement before retrieval or action, API-key and secrets management, trusted ingestion, and rate limiting
- Scalability and cost: concurrency handling, caching, latency budgets under load, and cost-per-interaction tracking
- Adversarial testing: malicious prompts, instructions hidden in ingested documents, poisoned records, sensitive-data extraction attempts, and cross-user access attempts
- *Hands-on:* Harden the multi-agent NorthBridge assistant into an operable production system.
 - Run the shared regression suite and agent-specific suites against a fixed corpus and configuration; publish a comparison report covering quality, faithfulness, task success, latency, and cost.
 - Add structured tracing for every triage decision, retrieved passage, tool call, agent handoff, approval gate, and failure reason; confirm secrets and sensitive content are redacted.
 - Add guardrails: prompt-injection test fixtures, PII redaction on outputs, and a tool allow-list enforced in code, not just in the prompt.
 -

Build isolated adversarial fixtures — malicious user prompts, poisoned policy documents, and cross-user record-access attempts — and test every agent and tool against them; remediate the highest-priority failures.

- Containerize the application; add environment validation, database and index provisioning, health and readiness checks, bounded retries and timeouts, and basic load testing for latency under concurrent requests.
- **Session checkpoint:** an evaluated, traced, guarded, and containerized multi-agent assistant with a documented security-test result set and a comparison report across configurations.

Session 5 — Client Delivery: Rollout, Adoption, and Capstone Review

Package the technical work into a deliverable a client stakeholder can approve, run, and adopt.

- Delivery fundamentals: deployment pack structure, operating guide, environment configuration, and update procedures for documents, records, and agent configuration
- Client-facing communication: architecture walkthrough, demo storytelling, objection handling, and communicating risk and uncertainty without losing stakeholder trust
- Rollout and adoption planning: pilot scope, success criteria, risk register, milestones, feedback loop, and a human-escalation path for what the system does not handle
- Production-readiness review: architecture, evaluation evidence, security-test results, remaining limitations, and go/no-go criteria
- Capstone review: presenting the solution the way a forward-deployed engineer presents it to a client sponsor
- *Hands-on:* Finish the capstone as a delivery, not just a build.
 - Assemble the FDE deployment pack in `docs/deployment-pack.md`: discovery summary, architecture, evaluation evidence, security results, operating guide, known limitations, and rollout plan with a risk register and milestones.
 - Run an end-to-end local production simulation from a clean setup, confirming health checks, provisioning, and the full triage ' retrieval/action ' escalation flow.
 - Prepare and deliver a stakeholder demo of the NorthBridge assistant to a peer acting as the client sponsor: architecture, live demo, evaluation and security evidence, risks, and the adoption plan.
 - Incorporate stakeholder feedback from the demo into the risk register and the deployment pack.
 - **Session checkpoint:** a deployed-or-locally-simulated, evaluated, secured, and documented capstone solution, delivered end to end through a stakeholder demo and a complete deployment pack.

Capstone project

Participants incrementally build the **NorthBridge Industrial Service-Desk Assistant**, a multi-agent enterprise AI solution that answers HR and IT policy questions, looks up employee and ticket records, and carries out approved actions (leave requests, ticket creation) against enterprise-style systems, escalating to a human whenever evidence or authority is insufficient. The capstone must be a complete, operational

solution meeting real-world delivery standards, with a reproducible path from a discovery workshop to a deployed, demoed system.

- **Source materials:** `raw/HR_Policy_Handbook.pdf` and `raw/IT_Support_Runbook.pdf` provide the policy content; `raw/PROVENANCE.md` records their origin and version. A mock HRIS and ticketing API, backed by SQLite, stand in for the enterprise systems a real deployment would integrate with.
- **Project and code ownership:** all implementation code belongs in `/Users/shakir/learning-capstones/forward-deployed-engineering-fde-2`, including ingestion, retrieval, agents, orchestration, the mock enterprise services, tests, evaluation tooling, and deployment configuration. `/Users/shakir/learning` holds course outlines and learning notes only.
- **Database and integration story:** a single MongoDB deployment serves as the application database, vector store, and agent-trace store, mirroring the pattern used elsewhere in Shakir's learning environment. The mock HRIS and ticketing APIs are a separate FastAPI/SQLite service, so participants build and test real API integration rather than reading enterprise data directly out of the vector store. Connection details load through environment variables or secret configuration; credentials never appear in course material, code, or logs.

Required competency coverage and implementation order

All six competency areas are mandatory in the capstone. Each must be demonstrated through the working application, not through a slide, a mock, or a notebook-only exercise.

Order	Required competency	Working capstone increment	Evidence required before completion
1	Enterprise AI Solution Design	Discovery notes, opportunity statement, architecture note, and evaluation baseline for NorthBridge.	Reviewed artifacts naming KPIs, integration points, and trust boundaries; a running skeleton with a health endpoint.
2	LLM & Agentic AI Expertise	Semantic-RAG assistant, then a bounded single agent, then a multi-agent system orchestrated with LangChain and LangGraph.	Working demo of triage ' retrieval/action ' escalation, with full step tracing and a single-agent vs. multi-agent comparison.
3	AI Deployment & Integration	Integration with the mock HRIS and ticketing APIs and the shared MongoDB database; containerized deployment.	Passing integration tests, a deployed or locally-simulated sandbox instance, and documented API/database contracts.

4	Production AI Experience	Evaluation harness, tracing/observability, guardrails, security controls, and load-tested scalability.	Comparison report, adversarial-test results, redacted traces, and latency/cost measurements under concurrent load.
5	Client-Facing Consulting Skills	Discovery workshop, stakeholder demo, and rollout/adoption plan with a risk register.	A delivered stakeholder demo, incorporated feedback, and a completed deployment pack.
6	Hands-on Project Experience	The complete NorthBridge assistant, working end to end from ingestion through action and escalation.	End-to-end demo tied back to the Day 1 KPIs, with an honest account of what improved and what did not.

One application, independently demonstrable configurations: the assistant must expose semantic-RAG-only, single-agent, and multi-agent modes so each can be compared on the same evaluation set rather than relying on the most complex mode to happen to be selected. Automated triage routing must log the selected path and its reason.

Learning and demonstration contract: for each increment, document the problem it addresses, its architecture and data flow, the integration points it touches, suitable and unsuitable use cases, and failure modes. Supply runnable code, automated checks, demonstration questions, inspectable traces, and a comparison report. Record improvements and regressions honestly; an increment need not outperform the previous one to be successfully implemented and understood.

Operational completion criteria: the capstone is complete only when its supported workflows run end to end against the shared MongoDB database and the mock HRIS/ticketing APIs, with a usable interface, grounded citations, traceable agent actions, and clear escalation behaviour. Completion requires automated ingestion, retrieval, agent, API, and end-to-end tests; documented evaluation thresholds and passing results; authorization and adversarial checks; protected secrets; structured logs and traces with sensitive-data redaction; bounded retries and timeouts; and a delivered stakeholder demo with a completed deployment pack. Any remaining limitation must be documented against the supported scope.

A note on delivery

This intensive covers Enterprise AI Solution Design, LLM & Agentic AI Expertise, AI Deployment & Integration, Production AI Experience, Client-Facing Consulting Skills, and Hands-on Project Experience across two sessions on Day 1, one session on Day 2, and two sessions on Day 3. The additional time on Day 2 reflects how much build work multi-agent orchestration requires; coverage is not thinned to fit the calendar.

The complete NorthBridge assistant is developed and validated in `/Users/shakir/learning-capstones/forward-deployed-engineering-fde-2` as we work through all six competency increments

together. Shakir's learning effort includes running the discovery workshop, building every agent and integration, and delivering the final stakeholder demo — not just watching demonstrations of each.

Each full day balances instructor-led explanation, demonstration, guided labs, review, and troubleshooting, with two tea breaks and one lunch break. A supplied application scaffold, mock enterprise APIs, policy documents, and evaluation questions keep the labs focused on solution design and agentic engineering rather than boilerplate. Participants arrive with the environment installed and verified so setup does not consume teaching time.

The sequence makes the dependencies explicit: discovery and architecture precede any build; a grounded retrieval baseline precedes agent design; a working single agent precedes multi-agent orchestration; and evaluation, guardrails, and security run alongside every increment rather than arriving only at the end. Participants leave with a working multi-agent enterprise assistant, comparison and security evidence, and a delivered client-facing rollout plan.

Ready to build and deliver enterprise AI solutions with engineering discipline?

Enquire about a private cohort or a tailored delivery for your forward-deployed engineering team.