

COURSE OUTLINE

GenAI Design & Architecture — Engineering Robust Enterprise AI Systems

An intensive architecture practicum for senior engineers designing governed, scalable GenAI products

DURATION

3 days

FORMAT

Architecture-led learning, case-study workshops, design reviews, and hands-on implementation exercises

DELIVERY

Online Instructor Led Workshop · Onsite Program for your Teams

TRAINER

Muhammed Shakir

Why this program exists

This program gives experienced engineers a shared architecture language and a practical decision framework for designing GenAI systems that can be governed and evolved in a large enterprise. It deliberately moves beyond model fundamentals and basic prompt engineering. Instead, participants work through the trade-offs that determine whether a GenAI proof of concept becomes a dependable product.

The learning journey follows one realistic enterprise copilot case study from retrieval design through agentic orchestration, evaluation, observability, security, and an executive-ready architecture review.

What makes this different

This is an architecture room: every technique is assessed in the context of an end-to-end system, its operating model, and its failure modes. Participants will:

- **Make architecture decisions, not just learn terminology** — use decision criteria to select retrieval, orchestration, memory, tool, and evaluation patterns.
- **Connect advanced patterns to production constraints** — address security, compliance, latency, cost, testability, resilience, and human accountability from the outset.
- **Work from a coherent enterprise case** — progressively design and stress-test a governed knowledge-and-action assistant rather than assemble disconnected demos.

•

Bring internal standards into the room — use selected workshops to map the patterns to your organisation's architecture principles, approved platforms, and internal AI guidance.

The program is vendor-neutral by design. Frameworks and model providers are treated as implementation choices beneath durable architectural principles.

Skills You'll Walk Away With

- Design an enterprise GenAI reference architecture with explicit trust boundaries, data flows, non-functional requirements, and operational ownership.
- Choose and justify retrieval patterns including embeddings, hybrid search, re-ranking, HyDE, query decomposition, GraphRAG, and agentic retrieval.
- Decide when a deterministic workflow is preferable to an agent, and apply prompt chains, ReAct-style loops, reflection, durable execution, and human-in-the-loop controls appropriately.
- Design safe agent building blocks: typed tools, scoped memory, sandboxed execution, MCP, A2A boundaries, and approval policies.
- Build an evaluation strategy spanning curated and synthetic datasets, automated and human evaluation, bias checks, regression testing, and online feedback.
- Instrument GenAI applications for traces, prompt/tool/retrieval visibility, quality, latency, reliability, token use, and cost.
- Apply practical controls for prompt injection, sensitive-data exposure, excessive agency, untrusted content, and unsafe tool use.
- Communicate an architecture recommendation and its trade-offs clearly to engineering, risk, product, and leadership stakeholders.

Who should attend

This program is for experienced technical professionals who are already contributing to GenAI initiatives and now need deeper system-design and engineering judgment.

- **Senior software engineers and technical leads** — designing and implementing GenAI-enabled services.
- **Solution, platform, and enterprise architects** — defining reusable patterns, integration boundaries, and guardrails.
- **ML/AI engineers and data engineers** — building retrieval, evaluation, and operational capabilities.
- **Engineering managers and staff engineers** — making delivery, reliability, and technology-selection decisions.
- **Security, quality, and platform specialists** — partnering with delivery teams on controlled adoption.

Prerequisites: participants should already be familiar with LLM application basics, APIs, prompting, and software delivery practices. Familiarity with one GenAI application or internal framework is useful. The program does not spend time teaching introductory model concepts.

Course Outline

Day 1 — Establish the Architecture Spine: Retrieval, Context, and Trust

Start from the business task and its constraints, then design the information and trust architecture that grounds every later decision.

- Enterprise GenAI architecture: system boundaries, data classification, trust zones, quality attributes, and failure budgets
- The context-engineering view: model context, application state, retrieved knowledge, instructions, tool results, and provenance
- Retrieval foundations revisited at architecture depth: chunking, metadata, embeddings, index design, freshness, tenancy, access control, and citations
- Retrieval pattern selection: semantic and hybrid search, re-ranking, HyDE, query decomposition, structured retrieval, GraphRAG, and agentic RAG
- When RAG is the wrong answer: direct system integration, workflow-owned data, search-only experiences, or clarification/escalation paths
- Threat modelling early: prompt injection, indirect prompt injection in documents, data leakage, retrieval poisoning, and access-control gaps
- *Architecture workshop*: Define the target enterprise copilot, its users, data domains, high-risk actions, quality attributes, and non-negotiable guardrails.
- *Hands-on lab*: Compare retrieval pipelines for representative questions; inspect relevance, provenance, permissions, latency, and failure behaviour, then record a pattern-selection decision.

Day 2 — Orchestrate Predictably: Workflows, Agents, Tools, and Memory

Turn a grounded assistant into a controlled system that can reason, act, recover, and ask for help without becoming unnecessarily autonomous.

- Workflow vs. agent: a practical decision matrix for determinism, risk, task variability, auditability, latency, and cost
- Orchestration patterns: prompt chaining, routing, planner/executor designs, ReAct, reflection, retry and repair loops, durable execution, and state machines
- Agent building blocks: structured outputs, typed tools, tool contracts, idempotency, validation, permissions, timeouts, and safe failure handling
- Memory architecture: session state, short-term working memory, long-term memory, retention policies, recall quality, and privacy boundaries
-

Sandboxed code execution and external actions: least privilege, isolation, approval gates, and audit trails

- MCP and A2A: what they standardise, where they fit, and how to define client/server or agent-to-agent trust boundaries
- Multi-agent systems: specialised roles, coordination costs, shared state, supervision, and the situations where a single orchestrator is better
- *Design clinic*: Convert three business scenarios into a deterministic workflow, a bounded agent loop, or a human-led process—and defend the choices.
- *Hands-on lab*: Design and implement an approved-tool interaction with structured inputs/outputs, confirmation policy, error recovery, traceability, and a deliberate adversarial test.

Day 3 — Prove It Can Operate: Evaluation, Reliability, Governance, and Architecture Review

Close the loop between design intent and measurable operating evidence—the discipline that enables responsible production adoption.

- Evaluation architecture: task taxonomy, golden datasets, synthetic test generation, LLM-as-judge, rubric design, bias mitigation, and human feedback loops
- Offline, pre-release, and online evaluation; A/B testing, regression gates, acceptance thresholds, and avoiding evaluator gaming
- Observability for GenAI: end-to-end tracing; prompt, retrieval, and tool logs; correlation IDs; quality signals; privacy-aware telemetry; and incident investigation
- Reliability and performance: model/provider fallbacks, retries, circuit breakers, graceful degradation, queueing, caching, rate limits, token budgets, and cost controls
- Governance by design: model selection, data residency, auditability, change management, policy-as-code, ownership, and architecture review evidence
- Internal adoption patterns: mapping the reference architecture to your organisation's approved frameworks, platform services, security controls, and AI operating model
- *Capstone architecture review*: Teams present an end-to-end design for the enterprise copilot: retrieval and orchestration choices, threat model, evaluation plan, telemetry, operating costs, and a phased delivery plan.
- *Facilitated critique*: Peer and instructor architecture review using a reusable scorecard; identify assumptions, risks, open decisions, and the next experiment needed to retire each risk.

Capstone case study

Across the three days, participants design a governed enterprise knowledge-and-action copilot. The system must answer from authorised internal knowledge, route requests through appropriate retrieval or workflow paths, use a carefully scoped external tool when justified, retain only approved state, and provide evidence for its answer and behaviour.

The deliverable is not a throwaway demo. Each team leaves with an architecture canvas, pattern-selection rationale, threat model, evaluation and observability plan, and a phased implementation roadmap that can be adapted to a real internal use case.

Recommended learning approach

The strongest outcome will come from combining shared architecture principles with your organisation's own constraints and examples.

- **Before the program:** nominate one or two suitably anonymised internal use cases; share relevant architecture principles, approved technology choices, security expectations, and current internal AI patterns where available.
- **During the program:** use short concept briefings, working labs, whiteboard/system-design exercises, and review conversations with internal SMEs at the points where company-specific choices matter.
- **After the program:** teams refine their capstone into a proposed pilot, including success metrics, evaluation data, controls, owners, and an architecture-review checkpoint.

This sequencing starts with trustworthy context, adds controlled orchestration only where it earns its complexity, and ends by proving the result can be measured and operated.

Trainer perspective and delivery confidence

The program is led by a trainer with **30 years of software engineering experience**, bringing a practitioner's view of architecture, delivery risk, and the organisational realities behind enterprise technology adoption. The facilitation style is deliberately senior-peer: direct, evidence-led, and anchored in decisions participants will need to make in their own systems—not generic tool demonstrations.

The trainer is also the creator of [Projekts AI](#), a platform for specification-led, AI-accelerated software delivery. Projekts connects shared product, architecture, UX, delivery, and engineering artifacts with model choice, cost visibility, and governance—practical experience that informs this course's focus on traceability, accountable handoffs, and production-ready GenAI architecture.

A note on delivery

Each day is structured as approximately six hours of focused in-person learning, balancing concise instruction, demonstrations, guided design, hands-on exercises, and architecture critique. The cohort size of approximately 15 is ideal for small-team workshops and meaningful review. The final agenda can be tailored around your organisation's internal frameworks and selected use cases without compromising the progressive learning journey.