

COURSE OUTLINE

AI for Developers — Build Practical, Responsible AI Features

A lab-driven foundation for developers moving from AI experimentation to working applications

DURATION

5 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

AI is becoming part of ordinary software delivery, but prompting a chatbot is not the same as engineering an AI feature that is useful, secure, testable, and maintainable. Developers need a clear model of what LLMs can do, where they fail, and how to put them behind sensible application boundaries.

This program turns AI from a black box into an engineering toolkit: participants use current AI tools and APIs to build and assess a practical knowledge-assistant application.

By the end of five days, participants will have moved from prompts and coding assistants to a small LLM application with one basic RAG example, one basic MCP example, evaluation checks, and production-minded safeguards.

What makes this different

Most introductory AI courses stop at concepts or isolated demos. **This is an engineering room: every important idea is connected to a small, inspectable system.** You will:

- **Learn AI in plain engineering language** — connect models, tokens, embeddings, tools, and agents to familiar ideas such as APIs, indexes, logs, tests, and feedback loops.
- **Use AI as a developer, not merely a consumer** — practise code generation, debugging, testing, refactoring, repository understanding, and human review with AI coding tools.
- **Build a right-sized working application** — extend a basic chat experience through one simple retrieval example and one controlled MCP tool example, then assess and package it responsibly.

The course favours clear mental models and small working examples over a rushed tour of advanced frameworks.

Skills You'll Walk Away With

- Explain the practical differences between AI, machine learning, generative AI, LLMs, and AI agents.
- Write and improve prompts using role, context, task, constraints, and examples.
- Use AI coding assistants responsibly for code generation, debugging, testing, refactoring, and review.
- Build a small LLM-backed chat application with streaming, error handling, API-key protection, and cost awareness.
- Describe and implement a basic RAG flow using a small document set and source-backed answers.
- Connect an LLM to one external capability through a simple, controlled MCP example with human oversight.
- Evaluate, secure, observe, and package an AI application using practical engineering checks.

Who should attend

This program is for technically minded early-career professionals who want to build AI-enabled software rather than only use AI chat tools.

- **College graduates** — preparing for AI-aware software development roles.
- **Junior developers** — adding AI application fundamentals to their engineering toolkit.
- **Frontend and backend developers** — integrating LLM capabilities into web applications and APIs.
- **QA and automation engineers** — learning how AI systems and AI-generated code should be assessed.
- **Technical career switchers** — with basic programming experience who need a practical entry to AI engineering.

Prerequisites: comfort with programming, HTTP APIs, Git, and basic command-line use. Deep machine-learning expertise or advanced mathematics is not required; the course starts with plain-language engineering models.

Course Outline

Day 1 — Demystify AI and Prompt with Intent

Build the vocabulary and prompt-design habits needed to make useful, realistic engineering decisions.

- AI vs. machine learning vs. deep learning; generative AI, LLMs, and AI agents
- Real-world use cases, limitations, multimodal AI, and the model landscape
- Tokens, context windows, training vs. inference, and their cost/quality implications

- Prompt anatomy: role, context, task, constraints, output format, and few-shot examples
- Prompt comparison and lightweight evaluation
- *Hands-on*: Design and compare prompts for a structured business task, recording output quality, latency, and token trade-offs.

Day 2 — Develop and Build with LLM APIs

Use AI coding tools responsibly, then turn a prompt experiment into a functioning application.

- Coding tools: GitHub Copilot, Cursor, Windsurf, Gemini Code Assist, Amazon Q, Claude Code, Codex, Cline, and Roo Code
- Code generation, explanation, debugging, testing, refactoring, repository context, and human review
- LLM APIs, SDKs, model selection, authentication, and chat-application patterns
- Streaming responses, error handling, retries, API-key protection, rate limits, and token costs
- *Hands-on*: Build a streaming chat application and use an AI coding assistant to add and review a tested feature.

Day 3 — RAG: One Clear Retrieval Example

Understand how an application can answer from supplied information without treating the model as a database.

- Embeddings, semantic search, documents, chunks, metadata, and vector stores—concepts before platform choice
- RAG flow: document ' embedding ' retrieval ' LLM answer
- Why retrieved context, citations, and answer boundaries matter
- **Basic RAG example (the single RAG build)**: load a small FAQ or policy document, retrieve the most relevant passage for one user question, and return an answer with its source
- Retrieval and answer checks: relevance, faithfulness, missing context, and hallucination
- *Hands-on*: Add the one document-Q&A RAG example to the chat application and test it with answerable and unanswerable questions.

Day 4 — MCP and Agentic Tool Use: One Controlled Example

Learn the difference between a model generating text and an application safely giving it access to a capability.

- AI assistants vs. agents; goals, tools, state, planning, and human-in-the-loop boundaries
- Function/tool calling, structured outputs, validation, permissions, and error recovery
- Model Context Protocol (MCP): a standard way for an AI client to discover and use approved tools or data sources
-

Basic MCP example (the single MCP build): connect the assistant to one read-only tool—such as a weather or company-directory lookup—through an MCP server; validate the request and show the returned result

- When multi-step or multi-agent designs help, and why they are not the default for a five-day foundation course
- *Hands-on:* Run, inspect, and test the single read-only MCP tool example with a human confirmation/validation step.

Day 5 — Evaluate, Secure, and Package the Application

Finish with the operating discipline that separates a useful demo from a responsible engineering starting point.

- Hallucination, accuracy, quality, latency, performance, and cost as measurable concerns
- Small evaluation datasets, rubrics, regression checks, and RAG/tool-use test cases
- Prompt injection, data leakage, sensitive data, API-key security, and reviewing AI-generated code
- Docker, environment configuration, logging, monitoring, caching/scaling concepts, and deployment readiness
- Capstone integration and architecture review: what the application does, what it must not do, and how it will be observed
- *Hands-on:* Package the knowledge assistant, run its prompt/RAG/MCP evaluation and security checklist, and present the resulting application and its safeguards.

Capstone project

Participants incrementally build a small knowledge assistant: a streaming LLM chat application, one FAQ/policy document RAG example, and one read-only MCP tool example. The capstone is intentionally bounded. It demonstrates the concepts clearly without claiming production-scale RAG, autonomous agents, multi-agent coordination, or a full enterprise deployment in five days.

A note on delivery

Each full day balances instructor-led explanation, demonstration, guided labs, review, and troubleshooting, with two tea breaks and one lunch break. The schedule moves from model and prompt fundamentals to application construction, then to one focused RAG example, one focused MCP example, and practical quality and delivery checks. Participants leave with working code and a dependable foundation for deeper RAG, agent, evaluation, and production training.

Ready to build AI features with engineering discipline?

Enquire about a private cohort or a tailored delivery for your developer team.