APPLIED AI ENGINEERING ACADEMY

LLM Engineering

From Prompting and Statistical Inference to Reliable RAG Systems

3 DAYS | 21 CONTACT HOURS | HANDS-ON INTENSIVE

A practical course for understanding how large language models behave, designing robust instruction and context layers, and building an evaluated evidence-grounded application.

AUDIENCE	DELIVERY	CAPSTONE
Engineers, data scientists, technical product leads and advanced AI practitioners.	One shared codebase developed through guided experiments, retrieval labs and evaluation.	An evidence-grounded assistant with hybrid retrieval, citations, tests and safety controls.

Feasibility verdict: Feasible as a working-proficiency course, provided GraphRAG indexing and prompt-program optimization are demonstrations rather than separate full projects.

Version 1.0 | Framework review date: 1 August 2026

Course overview

The course connects model mechanics, statistical inference, prompt and context engineering, retrieval, security and evaluation. Participants leave with a realistic mental model of LLMs and a small application that can be measured rather than merely demonstrated.

Learning outcomes

- Explain tokenization, embeddings, attention, next-token prediction, context windows, training and inference at an engineering level.
- Interpret logits, softmax probabilities, entropy, temperature, top-p sampling and the practical limits of probabilistic generation.
- Design system, developer, user and tool instruction layers, including typed and schema-constrained outputs.
- Build a conventional RAG pipeline with ingestion, chunking, dense and lexical retrieval, metadata filtering, reranking and citations.
- Decide when GraphRAG, long-context prompting, database queries or fine-tuning are more appropriate than basic RAG.
- Create an evaluation dataset and measure retrieval quality, groundedness, citation validity, latency and cost.
- Recognize direct and indirect prompt injection, retrieval poisoning, sensitive-data leakage and unsupported claims.

Realistic depth within three days

IMPLEMENT	PRACTISE	SURVEY / DEMO
• Prompt contract with typed output • Baseline and hybrid RAG pipeline • Small automated evaluation suite	• Sampling and failure-mode experiments • Chunking, retrieval and reranking • Prompt-injection and grounding tests	• Full transformer mathematics and training • GraphRAG indexing at scale • DSPy optimization and fine-tuning workflows

Conditions that make the schedule feasible

- Use one primary provider API and one prepared Python environment; provider comparisons are conceptual rather than repeated implementations.
- Supply a curated document corpus, a starter ingestion pipeline and an initial question set before the workshop.
- Pre-build the GraphRAG index; participants query and compare it instead of spending hours on extraction and indexing.
- Limit the capstone to a small domain assistant rather than a complete production platform.
- Target 12-20 participants; add a teaching assistant above 16 participants to keep labs moving.
- Allocate approximately 9.5 hours to guided coding and evaluation, 9 hours to concepts and demonstrations, and 2.5 hours to review and presentations.

Recommended participant profile

Primary audience. Software engineers, data scientists, ML practitioners, technical product managers, researchers and advanced business users who need to design or assess LLM applications.

Prerequisites. Basic Python, JSON and API familiarity. No prior deep-learning coursework is required, but participants should be comfortable reading short code examples and simple probability notation.

Day 1 - How LLMs Work and How to Instruct Them

Build an accurate mental model before learning prompt patterns.

Time	Module	Content and activity
09:00-09:30	**Orientation and baseline**	Course architecture, diagnostic task and the distinction between a model, an application and an agent.
09:30-10:30	**Model mechanics**	Tokens, embeddings, transformer blocks, self-attention, context windows, layers, logits, training versus inference.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**Statistical inference lab**	Conditional probability, softmax, entropy, temperature, top-p and sampling. Participants inspect distributions and output variance.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:15	**Capabilities and limitations**	In-context learning, reasoning, memorization, uncertainty, hallucination, long context and sensitivity to wording.
14:15-15:00	**Instruction hierarchy**	System, developer, user, assistant and tool messages; conflicts, delimiters, trust boundaries and structured outputs.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:15	**Prompt engineering**	Objectives, constraints, examples, rubrics, decomposition, few-shot patterns, output schemas and error recovery.
16:15-17:30	**Lab 1: prompt contract**	Create a typed prompt contract, test it against adversarial and ambiguous examples, and record failure categories.

DAY OUTPUT

A reusable prompt specification, a 20-case test set and a short catalogue of observed failure modes.

INSTRUCTOR NOTE

Do not attempt a full derivation of transformer training. Use visual intuition and small numerical experiments so enough time remains for application engineering.

Day 2 - Retrieval, RAG and Graph-Enhanced Context

Move from isolated prompting to evidence-grounded applications.

Time	Module	Content and activity
09:00-09:30	**Recap and retrieval diagnostic**	Review Day 1 and demonstrate why an LLM without retrieval cannot reliably answer corpus-specific questions.
09:30-10:30	**Search foundations**	Embeddings, cosine similarity, BM25, dense versus sparse search, metadata filters and hybrid retrieval.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**RAG pipeline**	Parsing, chunking, overlap, metadata, indexing, query rewriting, candidate retrieval and reranking. Guided implementation.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:15	**Grounded generation**	Context construction, citation design, abstention, source trust, context budgets and answer synthesis.
14:15-15:00	**GraphRAG**	Entities, relations, communities, global and local questions, indexing cost and decision criteria for graph-based retrieval.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**RAG security**	Indirect prompt injection, poisoned documents, access control, sensitive data, malicious metadata and trust separation.
16:00-17:30	**Lab 2: retrieval comparison**	Build baseline dense retrieval, add lexical fusion and reranking, then compare with a prepared GraphRAG index.

DAY OUTPUT

A working RAG pipeline and a retrieval comparison report covering recall, precision, latency and answer grounding.

INSTRUCTOR NOTE

GraphRAG is queried and evaluated, not indexed from scratch. This preserves at least 90 minutes for participants to improve conventional retrieval, which is more broadly applicable.

Day 3 - Evaluation, Security and Production Readiness

Turn a convincing demonstration into a testable engineering system.

Time	Module	Content and activity
09:00-10:00	**Evaluation design**	Golden datasets, deterministic assertions, semantic grading, human review, model judges, pairwise comparison and leakage risks.
10:00-10:30	**RAG metrics**	Retrieval recall, context precision, groundedness, completeness, citation validity, latency and cost.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**Lab 3: evaluation pipeline**	Implement automated checks, run regressions and analyse failure clusters across prompt and retrieval versions.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:00	**Production architecture**	Model selection, routing, caching, fallbacks, rate limits, privacy, long context versus RAG and fine-tuning decisions.
14:00-15:00	**Observability and operations**	Prompt and model versions, traces, retrieval diagnostics, feedback, drift, cost budgets and release gates.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**Advanced optimization demo**	DSPy signatures, modules, metrics and optimization as a transition from hand-edited prompts to measurable programs.
16:00-17:00	**Capstone hardening**	Add injection tests, uncertainty behaviour, citation checks, monitoring fields and an architecture decision record.
17:00-17:30	**Demonstrations and review**	Short team demonstrations, trade-off discussion and next-step recommendations.

DAY OUTPUT

An evaluated, red-teamed domain assistant with a concise architecture and operations report.

INSTRUCTOR NOTE

DSPy remains a demonstration unless the group is already experienced. The course objective is a reliable RAG application, not mastery of every optimization framework.

Hands-on structure and capstone

Progressive lab sequence

Lab	Focus	Participant deliverable
1	Prompt laboratory	Typed prompt contract, structured output schema, test cases and failure-mode log.
2	Evidence-grounded retrieval	Dense baseline, hybrid search, reranking, citations and a prepared GraphRAG comparison.
3	Evaluation and red team	Automated regression report, grounding checks, injection cases, latency and cost measurements.
4	Capstone integration	A coherent assistant combining the previous labs with documentation and a release decision.

Capstone specification

Teams build a reliable domain knowledge assistant over a supplied corpus. The assessment focuses on evidence, measurable behaviour and explicit failure handling rather than visual polish.

- Separate developer instructions from untrusted user and retrieved content.
- Typed input and output models with validation and graceful error handling.
- At least two retrieval signals, such as dense and lexical retrieval, followed by reranking.
- Source citations, uncertainty or abstention behaviour, and unsupported-claim detection.
- A minimum 20-case evaluation set including normal, ambiguous and adversarial cases.
- Measured retrieval quality, groundedness, latency and approximate cost.
- A one-page architecture decision record explaining why conventional RAG, GraphRAG or another approach was selected.

Assessment

Component	Weight
Concept checks and daily reflection	15%
Three guided lab deliverables	35%
Final capstone system and evaluation evidence	35%
Technical explanation and trade-off defence	15%

Framework stack and delivery operations

Primary workshop stack

Layer	Recommended framework	Why it is used in this course
Model access	**One provider SDK**	Keeps attention on model behaviour and application design rather than cross-provider syntax.
Schemas	**Pydantic**	Typed inputs and structured outputs make prompt contracts testable.
RAG orchestration	**Haystack**	Explicit modular pipelines are well suited to teaching ingestion, retrieval, reranking and generation.
Retrieval	**Qdrant or PostgreSQL + pgvector**	Qdrant is convenient for hybrid-search labs; pgvector is useful when relational data should remain in PostgreSQL.
Graph retrieval	**Microsoft GraphRAG**	Used as a prepared comparison for entity, relationship and corpus-level questions.
Prompt programs	**DSPy**	Demonstrates metric-driven optimization beyond manually editing prompt strings.
Testing	**pytest + task-specific evaluators**	Combines deterministic assertions with model- and retrieval-level evaluation.

Frameworks discussed but not all taught hands-on

- LlamaIndex can replace Haystack when data connectors and indexing abstractions are the primary concern; teaching both would dilute the workshop.
- Managed vector databases may replace Qdrant or pgvector in production, but the workshop keeps infrastructure transparent.
- Long-context prompting, SQL or graph queries and fine-tuning are treated as alternative architectural choices, not automatic upgrades to RAG.

Operational requirements

PARTICIPANT SETUP

Laptop with Python 3.11+, Git, a modern editor and access to the supplied repository. API credentials or a workshop gateway are provided by the organizer.

INSTRUCTOR PREPARATION

Pin all packages, pre-index the corpus and GraphRAG dataset, pre-run every notebook, provide fallback cached outputs and test quota limits.

CLASS SIZE

12-20 participants per instructor. Add one teaching assistant above 16 participants or when Python experience varies significantly.

DATA AND SAFETY

Use synthetic or public documents. Do not connect the exercises to confidential corpora or production accounts during the workshop.

Framework versions should be pinned and the complete lab environment should be tested immediately before delivery. The framework recommendations reflect documentation available on 1 August 2026.

Selected current references

Official documentation and standards used to validate the framework recommendations and the current syllabus scope. Accessed 1 August 2026.

1. Text generation and prompting guides — *OpenAI Developers.* Current API guidance for text generation, prompting and structured output.
2. DSPy documentation — *Stanford NLP.* Signatures, modules, metrics and prompt-program optimization.
3. Haystack documentation — *deepset.* Component-based RAG, agent and search pipelines.
4. Hybrid search and reranking — *Qdrant.* Dense and sparse candidate fusion followed by reranking.
5. GraphRAG documentation — *Microsoft Research.* Graph indexing and global, local and DRIFT search modes.
6. pgvector — *pgvector project.* Vector similarity search within PostgreSQL.

Syllabus maintenance note

Before each delivery, revalidate provider message-role semantics, structured-output APIs, supported embedding and reranking models, framework installation instructions and any GraphRAG configuration changes.

> **Important:** The course teaches working proficiency. It does not claim that participants can train a foundation model, build a large enterprise knowledge graph or certify a production system after three days.