APPLIED AND GENERATIVE AI

Evaluating and Monitoring Generative AI Applications

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Most teams ship LLM features on vibes: someone tries a few prompts, the answers look good, and it goes to production. Then a model update, a prompt tweak, or an unusual user quietly breaks quality, and nobody notices until users do. Traditional testing does not transfer cleanly, because LLM outputs are nondeterministic and often have no single right answer. Measuring quality is the genuinely hard problem at the center of running generative AI in production.

This is a hands-on, practitioner course. It builds the discipline from the ground up: first what quality even means for your application and why measuring it is hard, then constructing a test set worth trusting, then the evaluation methods themselves, from code-based checks to LLM-as-judge to human review. With measurement in place, the course moves to where it pays off: evals in the development loop, observability in production, and feedback loops that improve the system over time. The course covers fewer techniques than a survey would and practices each until it sticks. Every module includes a lab, and each module builds on the one before it.

Who Should Attend

- Developers and AI engineers who own the quality of an LLM feature in production
- QA and platform engineers extending their testing practice to generative AI
- Technical leads who need evidence, not anecdotes, before shipping AI features

Prerequisites

- Experience building or working on an application that calls an LLM API
- Working proficiency in Python
- Basic familiarity with automated testing concepts

  Learners who have not yet built anything with an LLM should start with *Building Generative AI Applications*.

What You Will Learn

- Explain why evaluating LLM applications is hard and what makes a metric trustworthy
- Build a representative test set with golden examples and meaningful edge cases
- Apply code-based checks, LLM-as-judge evaluation, and human review, and know when each fits
- Run evaluations in the development loop to compare prompts and models and catch regressions
- Instrument a production LLM application: logging, tracing, cost, latency, and quality signals
- Design feedback loops that turn production data into a steadily improving system

Course Outline

Day one: learning to measure quality

- Why Evaluation Is Hard
  - Nondeterminism, subjectivity, and the failure of vibes-based development
  - Defining quality for your application: correctness, groundedness, tone, safety, format
  - What a good metric looks like, and common metrics that mislead
  - Lab: hand-score a set of real LLM outputs, compare judgments, and surface disagreement
- Building a Test Set
  - Collecting representative examples from real usage and from imagination
  - Golden answers, rubrics, and cases with no single right answer
  - Coverage: happy paths, edge cases, and adversarial inputs
  - Lab: build a first test set for a working LLM application and defend its coverage
- Evaluation Methods
  - Code-based checks: exact match, assertions, and structural validation
  - LLM-as-judge: prompting a model to grade, and calibrating it against humans
  - Human review: where it is irreplaceable and how to spend it wisely
  - Lab: implement all three methods against the same test set and compare their verdicts

Day two: evaluation and monitoring in the lifecycle

- Evals in the Development Loop
  - Regression suites: catching quality drops from prompt and model changes
  - A/B comparison of prompts, models, and parameters with evidence
  - Wiring evals into CI without slowing the team down
  - Lab: catch a deliberately introduced prompt regression with an automated eval run
- Production Observability
  - Logging and tracing LLM calls: inputs, outputs, tokens, cost, and latency
  - Dashboards and alerts: which signals predict quality problems
  - Sampling production traffic for ongoing quality review
  - Lab: instrument an LLM application and build a dashboard of its health signals
- Feedback Loops and Continuous Improvement
  - Capturing user feedback, explicit and implicit
  - Detecting drift when models, data, or users change
  - Triage: turning production failures into new test cases
  - Lab: close the loop by promoting real production failures into the regression suite

Extended Version

The three-day version keeps the same gradient and adds depth where teams need it most:

- Evaluating retrieval-augmented systems, connecting to *Retrieval-Augmented Generation (RAG) with Vector Databases*
- Evaluating agentic behavior: multi-step tasks, tool use, and trajectory quality
- Red-teaming and safety evaluation, including prompt injection probing

- A capstone that builds a complete evaluation and monitoring harness for a working application