

AI & Generative AI App Development

Build useful, evaluated and responsible applications on large language models with Python

AI and Automation · intermediate · Live online, instructor-led

This syllabus is a draft awaiting instructor review and may change.

A twelve-week live course in building LLM applications with Python: LLM APIs, prompt design, structured outputs, embeddings, vector search, retrieval-augmented generation, evaluations, safety and deployment.

Length and schedule

48 hours of live teaching, on either schedule:

- Weekday Intensive (Mon–Fri): about 10 weeks before holidays
- Weekend Evening (Sat & Sun evenings): about 12 weeks before holidays

Fees per student, whole course

- Group: Rs. 14,999
- Duo: Rs. 24,999
- One-to-One: Rs. 44,999

Launch fee — GST treatment to be confirmed.

What you will be able to do

- Write clean Python that calls web APIs, handles errors and keeps secrets out of code
- Explain how large language models work, what they are good at and where they fail
- Call an LLM API with streaming, retries, timeouts and cost tracking
- Design, test and iterate prompts that behave predictably on real inputs
- Get validated structured outputs from a model and handle the cases where validation fails
- Build semantic search with embeddings and a vector index
- Build a retrieval-augmented generation pipeline that cites its sources and refuses when it lacks evidence
- Evaluate an LLM application with a test set and catch regressions before release
- Defend against prompt injection and data leakage, and apply privacy, consent and cost limits
- Deploy an AI application with configuration, logging and monitoring suitable for real users

Before you start

- Basic programming in any language: variables, conditions, loops and functions (Python is refreshed in the first module)
- Comfortable using a terminal to run commands and navigate folders
- Basic familiarity with JSON and how web APIs are called is helpful but not required
- A laptop with at least 8 GB RAM (16 GB recommended for Docker) running Windows, macOS or Linux
- Access to an LLM API account for practice; usage costs, free tiers and the provider chosen for the batch are explained before it starts

Tools

Python 3.12 or later, Visual Studio Code and Jupyter notebooks, An LLM API (provider chosen per batch), An embeddings model (hosted or open-source, chosen per batch), Pydantic, FastAPI, PostgreSQL with pgvector, pytest, Docker, Git and GitHub

Syllabus

Module 1: Python foundations for AI applications

The Python the rest of the course relies on: data structures, functions, files and JSON, calling web APIs and handling configuration and secrets safely.

1. Course setup and Python essentials (2 h)
2. Functions, data structures and JSON (2 h)
3. Calling web APIs and handling secrets (2 h)
4. Python review and doubt-clearing (2 h)

Module 2: LLM APIs and prompt design

How large language models behave, how to call them reliably from code and how to write prompts that produce predictable results.

5. How large language models work (2 h)
6. Calling an LLM API from Python (2 h)
7. Prompt design fundamentals (2 h)
8. Prompt lab and doubt-clearing (2 h)

Module 3: Structured outputs and AI-powered backends

Get typed, validated data back from a model and wrap LLM features in a small, well-behaved web service.

9. Structured outputs with schemas and Pydantic (2 h)
10. Extraction and classification pipelines (2 h)
11. Building an AI backend with FastAPI and doubt-clearing (2 h)

Module 4: Embeddings and vector search

Represent text as embeddings, store them in a vector index and build semantic search that finds meaning, not just keywords.

12. Embeddings and semantic similarity (2 h)
13. Vector search with PostgreSQL and pgvector (2 h)
14. Search lab and doubt-clearing (2 h)

Module 5: Retrieval-augmented generation

Ground model answers in your own documents: ingestion, chunking, retrieval, reranking and answers that cite their sources.

15. RAG architecture and document ingestion (2 h)
16. Improving retrieval quality (2 h)
17. Grounded answers, citations and permissions (2 h)
18. RAG review and doubt-clearing (2 h)

Module 6: Evaluation, safety and responsible AI

Measure answer quality instead of trusting it, and protect users, data and budgets from the ways LLM applications go wrong.

19. Evaluating LLM applications (2 h)

- 20. Security and safety for LLM applications (2 h)
- 21. Privacy, consent, cost and human oversight (2 h)

Module 7: Deployment and final project

Package and deploy an AI application with monitoring and cost controls, then complete, present and defend the final project.

- 22. Deploying and operating AI applications (2 h)
- 23. Final project build lab and mid-point review (2 h)
- 24. Final project presentations and final assessment (2 h)

Final project

Build, evaluate and deploy a grounded document assistant. A small organisation — for example a training centre, a clinic or a housing society — wants staff to get answers from its own policies, handbooks and FAQs without searching through files. You receive a written brief with sample documents, user stories and acceptance criteria, and you build it individually or in a pair. You may substitute your own document set if it contains no personal data you do not have permission to use. Deliverables- An ingestion pipeline that loads, cleans and chunks the documents, creates embeddings and stores them in PostgreSQL with pgvector, with metadata for source and access level.- A FastAPI service with an endpoint that answers questions using retrieval-augmented generation, cites the passages it used and says clearly when the documents do not contain the answer.- At least one structured-output feature validated with Pydantic, for example extracting key dates and obligations from a policy into a typed record.- An evaluation set of at least thirty questions with expected answers or sources, an automated evaluation script, and a short report comparing two retrieval or prompt configurations.- Safety measures: a documented prompt-injection test with the defences you added, personal-data redaction before anything is logged, and access filtering so a user only retrieves documents they are allowed to see.- Cost controls: a per-request token limit, a daily budget or rate limit, and a log of usage per request.- A Dockerfile, environment-based configuration with no secrets in the repository, and a deployment to a cloud environment or a documented local deployment.- A README with setup steps, an architecture diagram, the evaluation results and a list of known limitations, including the kinds of questions the assistant should not be trusted with. The project is presented and reviewed live in the final week, with feedback on answer quality, evaluation method, safety and design decisions.

Certificate

- At least 80% attendance
- All assignments submitted
- Projects passed
- Final assessment passed (60%)

A ZORK Academy course completion certificate; paying for the course does not by itself earn it.

Next step

Book a free consultation: <https://zorkacademy.online/consultation?course=ai-generative-ai-app-development>

We help you prepare for work; we do not guarantee jobs, placements or salaries.