GOVARDHAN INSTITUTE

Generative AI & Agentic AI

Course Syllabus

From Python Basics to Production-Ready AI Agents

Course Overview

This course takes learners from a Python foundation all the way to building, evaluating, and deploying production-grade Generative AI and Agentic AI systems. Each module builds on the previous, combining theory with hands-on demos and real-world projects.

Course Highlights

- 24 structured modules from beginner to advanced
- Hands-on demos in every module using real AI tools
- Covers the full stack: Python → LLMs → RAG → Agents → Deployment
- Uses industry tools: LangChain, LangGraph, LangSmith, FAISS, Ollama, FastAPI, Streamlit
- Ends with a Capstone Project — build and demo a production AI app

Module Breakdown

#	Module	Topics Covered
01	**Python Recap**	• Data structures, functions, OOP refresher • File handling, virtual environments, pip & requirements • Async Python — asyncio basics (critical for agentic workflows) • Type hints and Pydantic basics
02	**API Development with FastAPI**	• REST API concepts, routing, request/response models • Pydantic validation and dependency injection • Building and testing AI-ready endpoints • Async endpoints and background tasks

03	**UI Development with Streamlit**	• Building interactive AI demo apps • Session state and multi-turn chat UI components • File upload, sidebar controls, progress indicators • Connecting Streamlit frontend to FastAPI backend
04	**Prompt Engineering**	• Core concepts: zero-shot, few-shot, chain-of-thought, role prompting • Advanced techniques: ReAct, self-consistency, prompt chaining, Tree of Thought • System prompts, persona design, and instruction tuning • Prompt injection attacks and mitigation strategies • Demo: Prompt Playground (ChatGPT / Claude / Gemini) • Prompt evaluation, iteration strategies, and A/B testing prompts
05	**LLM Fundamentals**	• Transformer architecture — conceptual overview • Tokenization, context window, attention mechanism • Types of LLMs: open-source vs closed, base vs instruction-tuned • Key providers: OpenAI, Anthropic, Google, Meta, Mistral • Hallucination, grounding, and model limitations
06	**Structured Output & Function Calling**	• JSON mode and enforcing structured schemas from LLM responses • Tool/function calling — concepts and patterns • Parsing and validating LLM output with Pydantic • Real-world use cases: form filling, data extraction, API orchestration
07	**Local LLMs with Ollama**	• Why local LLMs? Cost, privacy, and offline use cases • Installing and running Ollama — Llama 3, Mistral, Phi, Gemma • Comparing local vs cloud model performance and tradeoffs • Integrating local models into Python apps via Ollama API
08	**Connecting with LLMs using Python (without LangChain)**	• OpenAI SDK and Anthropic SDK — setup and authentication • REST calls via requests / httpx • Streaming responses, token counting, error handling and retries • Cost tracking and rate limit management
09	**LangChain Introduction**	• Why LangChain? Architecture overview and core philosophy • Core components: chains, prompts, output parsers, memory

		• LangChain vs LlamaIndex vs raw SDK — when to use what • Setting up LangChain project structure
10	**Connecting with LLMs using LangChain**	• ChatModels, PromptTemplates, OutputParsers • LCEL — LangChain Expression Language basics • Switching between providers: OpenAI, Anthropic, Ollama, Google • Managing credentials and environment configs
11	**LLM Parameters Deep Dive**	• Temperature, Top-K, Top-P, Max Tokens, Stop Sequences • Frequency penalty and presence penalty • Seed-based determinism and reproducibility • Hands-on: observing output changes across parameter settings
12	**LangChain Pipelines — LCEL**	• Building modular pipelines with the | operator • Branching, fallbacks, and parallel chains • Runnable sequences and composition patterns • Error handling and retries inside pipelines
13	**Demo — Conversational AI**	• Multi-turn conversation design principles • Memory types: buffer, summary, window, entity memory • Building a full chatbot end-to-end (Streamlit + LangChain + LLM) • Adding streaming responses and typing indicators to UI
14	**RAG — Retrieval-Augmented Generation**	• Why RAG? Limitations of LLMs without external context • Text chunking strategies: fixed, recursive, semantic, hierarchical • Embeddings — what they are and how they work • Vector Databases deep dive: ChromaDB, FAISS, Pinecone, Weaviate, Qdrant • Indexing strategies, metadata filtering, namespace management • Retrieval strategies: similarity search, MMR, hybrid search (BM25 + dense) • Advanced RAG: HyDE, re-ranking, self-querying, multi-query retriever • RAG evaluation with RAGAS framework — faithfulness, relevance, groundedness • Demo: Document Chatbot with source citations
15	**Multimodal LLMs**	• What are multimodal models? Image + text, audio, video inputs • Key models: GPT-4o, Gemini 1.5 Pro, Claude 3, LLaVA (open-source) • Use cases: document understanding, image Q&A, screenshot parsing

		• Building a multimodal app — image upload + LLM response • Multimodal RAG — embedding and retrieving images
16	**Fine-Tuning LLMs**	• When to fine-tune vs RAG vs prompt engineering — decision framework • Dataset preparation, formatting, and quality guidelines • Fine-tuning approaches: full fine-tune, LoRA, QLoRA • Tools: Hugging Face Trainer, Unsloth, OpenAI fine-tune API • Evaluation of fine-tuned models vs base models
17	**Agentic AI — Theory & Patterns**	• What is an agent? Perception → Reasoning → Action loop • Single-agent architecture — components and lifecycle • Multi-agent systems: roles, communication protocols, coordination • Agentic patterns: ReAct, Plan-and-Execute, Reflection, Self-Critique, Debate • Tool use and function calling in agentic context • Human-in-the-loop checkpoints and intervention design • Long-term memory architectures for agents (Mem0, Zep)
18	**LangChain Agents — Implementation**	• LangChain AgentExecutor, tools, and toolkits • LangGraph — stateful multi-agent workflows and state machines • Building custom tools and tool validation • Supervisor agents, subgraph patterns, conditional routing • Memory management in multi-agent systems • Demo: Multi-agent research assistant
19	**LLMOps**	• ML lifecycle adapted for LLM applications • Prompt versioning — treating prompts like code (version control, rollback, A/B) • Experiment tracking: LangSmith, MLflow, Weights & Biases • Monitoring in production: latency, cost, token usage, failure rates • Semantic caching to reduce cost and latency (GPTCache, Redis) • Model swapping without breaking the application • Feedback loops — collecting user signals to improve the system • Observability and tracing with OpenTelemetry for LLMs
20	**Evaluation of Agents & LLM Apps**	• Evaluation frameworks: LangSmith, DeepEval, PromptFoo • Metrics: faithfulness, relevance, groundedness, task completion rate

		• Unit testing agents, tools, and pipelines • Human evaluation vs automated eval — when to use each • Regression testing — detecting quality degradation after updates • Building eval datasets and golden answer sets
21	**MCP — Model Context Protocol**	• What is MCP and why it matters for the AI ecosystem • MCP architecture: hosts, servers, and clients • Building MCP servers and exposing tools/resources • Connecting MCP tools into LangChain and agentic workflows • MCP vs function calling — comparison and tradeoffs
22	**Deployment**	• Containerization with Docker — Dockerfile for LLM apps • Deploying on cloud: AWS / GCP / Azure — overview and options • Serving LLM apps: FastAPI + Uvicorn, Streamlit Cloud, HuggingFace Spaces • Scalability, load balancing, and semantic caching with Redis • CI/CD pipelines for AI applications • Environment management: secrets, configs, .env best practices
23	**AI Governance, Security & Human-in-the-Loop**	• AI risk categories and responsible AI principles • Prompt injection, jailbreaking, and data leakage — attack vectors • Guardrails: NeMo Guardrails, Guardrails AI • PII handling, data masking, and privacy-by-design • Human-in-the-loop design patterns for critical workflows • Compliance basics: GDPR, EU AI Act overview • Model cards, audit trails, and explainability
24	**Capstone Project**	• End-to-end project combining: RAG + Agents + Deployment + Eval • Teams build a production-ready AI application from scratch • Project ideas: AI research assistant, customer support agent, document analyst • Code review, architecture walkthrough, and live demo • Peer evaluation and instructor feedback session

Tools & Technologies

Category	Tools
Languages & Frameworks	Python 3.11+ • FastAPI • Streamlit • LangChain • LangGraph
LLM Providers	OpenAI (GPT-4o) • Anthropic (Claude) • Google (Gemini) • Meta (Llama 3 via Ollama) • Mistral
Vector Databases	ChromaDB • FAISS • Pinecone • Weaviate • Qdrant
Eval & Observability	LangSmith • RAGAS • DeepEval • PromptFoo • Weights & Biases
Infrastructure	Docker • Redis • AWS / GCP / Azure • CI/CD (GitHub Actions)
Agentic & MCP	LangGraph • Ollama • MCP (Model Context Protocol) • Mem0 / Zep

Prerequisites

- Basic Python programming knowledge
- Familiarity with REST APIs is a plus
- No prior ML / AI experience required