AI Native Development

Duration:

30 Hours

Course Overview:

This advanced, hands-on specialization equips software developers, QA engineers, DevOps professionals, and technical architects to design, build, and operate **AI-native applications** using Large Language Models (LLMs), Agents, Langchain, retrieval-augmented generation (RAG), Model Context Protocol (MCP), agentic frameworks, guardrails and observability.

The course focuses on **real-world, production-grade AI systems**, covering Python essentials, LLM APIs, agent design, controlled context sharing with MCP, knowledge grounding, observability, guardrails, and multi-agent orchestration using OpenAI / Gemini models, LangChain, LangGraph, and modern AI application patterns.

Course Objectives:

Upon completion, participants will be able to:

- Build AI-native backend services using Python, FastAPI, and LLM APIs
- Design and implement AI agents using LangChain fundamentals
- Build reliable Retrieval-Augmented Generation (RAG) systems
- Apply Model Context Protocol (MCP) for secure, controlled context sharing
- Implement guardrails, observability, and evaluation for AI systems
- Design and orchestrate multi-agent workflows using LangGraph

Pre-requisites:

- Basic programming experience in any modern language (Python, Java, C#, JavaScript)

- Familiarity with REST APIs and backend development concepts

- Exposure to GitHub, CI/CD, or DevOps workflows is beneficial

- Prior completion of AI First / AI Literacy training is strongly recommended

Course Outline:

Module 1: Python Essentials for AI-Native Development

- Why Python dominates the AI ecosystem
- Python mental model for non-Python developers
- Essential Python constructs for AI applications
- Lists, dictionaries, and JSON handling
- Writing clean, readable Python for AI workloads
- Virtual environments and dependency management
- Async basics and why they matter for LLM calls
- Structuring AI-first Python projects

Demo/Hands-on:

- Setup lab environment or connect to cloud labs
- Set up Python environment and run python examples
- Explore Python advanced features for AI-Native Development

Outcome:

- Set up Python environment and run python examples and get ready for AI-Native dev

---

Module 2: LLM Foundations & Interaction Patterns

- From traditional APIs to LLM-powered services
- How LLMs interpret instructions and context
- Tokens, context windows, and temperature
- Closed vs Proprietary, OpenAI, Anthropic and Gemini models overview
- Chat vs completion interaction patterns
- Standard outputs and response validation
- Streaming responses and real-time interaction patterns
- Cloud vs local LLMs: when and why
- Introduction to Ollama for local LLM hosting

Demo/Hands-on:

- Setup and create Gemini / OpenAI API Key
- Call OpenAI / Gemini APIs (blocking and streaming)
- Run a local LLM with Ollama, and compare latency, cost, and output quality.

Outcome:

- Participants understand LLM behaviour, consumption patterns and deployment options

---

Module 3: LLM Services, FastAPI & Frontend Integration

- Treating LLMs as backend services
- AI service architecture patterns
- Introduction to FastAPI for AI applications
- Sync vs async endpoints
- Model abstraction and configuration-driven design
- Exposing streaming endpoints
- API design for AI-powered frontends
- Agent and chat UI interaction patterns
- Backend vs frontend responsibilities

Demo/Hands-on:

- Build an LLM-powered FastAPI backend
- Expose streaming endpoints
- Connect a simple frontend client (Streamlit)

Outcome:

- Participants can design and deploy LLM-backed backend services with clean abstractions

---

Module 4: LangChain Fundamentals – Building AI Agents

- What is an AI agent and when to use one
- LangChain mental model and architecture
- Chains vs agents
- LangChain concepts and features
- LLM integration and hyper parameters
- Handling streaming responses
- Exposing as Fast API service

Hands-on:

- Build a LangChain agent with LLM integration
- Experiment with hyper parameters
- Build a LangChain agent with streaming response
- Expose as FastAPI service and access from UI

Outcome:

- Participants understand and build practical AI agents with Langchain

---

Module 5: LangChain Advanced - Practical Agent Design & Composition

- Tools and tool calling concepts
- Basic memory concepts
- Human-in-the-loop patterns
- Multi-step chains using LCEL
- Prompt templates and reuse
- Conversation memory vs task memory
- Error handling and fallback strategies
- Designing maintainable agent pipelines

Hands-on:

- Build a LangChain agent capable of reasoning and tool calling
- Build multi turn conversational agents
- Refactor agents using LCEL with multi step workflows
- Expose as FastAPI service and access from UI

Outcome:

- Participants build clean, maintainable multi step agent workflows using Langchain

---

Module 6: Retrieval-Augmented Generation (RAG) Fundamentals

- Why RAG is essential for enterprise AI
- RAG architecture and core components
- Embeddings and semantic similarity
- Chunking strategies (what actually works)
- Vector database concepts
- Building RAG ingestion pipeline using LangChain

Hands-on:

- Setup and connect to Vector DB
- Build a document ingestion pipeline and store embeddings
- Explore the stored chucks in the Vector DB

Outcome:

- Participants can ground AI responses in enterprise data

Module 7: Practical RAG Design & Integration

- Retrieval patterns and relevance tuning
- Metadata filtering
- Common RAG failure modes
- Integrating RAG with agents
- When not to use RAG
- Build RAG retrieval and search pipeline using Langchain

Hands-on:

- Integrate RAG into an existing agent and improve answer quality
- Expose as FastAPI service and access from UI

Outcome:

- Participants design reliable, business-ready RAG systems.
- Expose as FastAPI service and access from UI

Module 8: Model Context Protocol (MCP)

- Why context management becomes critical at scale
- Limitations of prompt-only context sharing
- Introduction to Model Context Protocol (MCP)
- MCP architecture and key concepts
- MCP vs RAG vs fine-tuning
- Standardising tool calling with MCP
- Stdio vs SSE vs Streamable HTTP
- Integrating MCP with agents
- How to build custom MCP servers

Hands-on:

- Integrate existing MCP server from Agents with stdio
- Integrate existing MCP server from Agents with Streamable HTTP
- Build custom MCP server

Outcome:

- Participants understand how MCP works, integrate with Agents and build custom MCP server

---

Module 9: Guardrails and Safety

- Guardrails overview and concepts
- Prompt injection and misuse risks
- Input and output validation
- Schema enforcement and fail-safe responses
- Safe tool execution
- Integrate with agents and tool calling

Hands-on:

- Add guardrails in the agent and tool calling
- Validate and tune the responses
- Integrate with agentic workflow and expose as Fast API service

Outcome:

- Participants design safer and more predictable AI systems

Module 10: Observability, Evaluation & Cost Tracking

- AI observability vs traditional logging
- Langsmith / Langfuse concepts and tracing
- Evaluating prompts, RAG, and agents
- Debugging AI behavior
- Continuous improvement loops
- Cost tracking and optimization
- Integration with Langchain

Hands-on:

- Setup and connect to observability platform (Langsmith / Langfuse)
- Instrument agents and compare different prompt and retrieval strategies
- Track token usage, latency and cost

Outcome:

- Participants design safer and more predictable AI systems

Module 11: Multi-Agent Systems

- When multi-agent systems are needed
- LangGraph concepts: nodes, edges, and state
- Planner–Executor–Reviewer pattern
- Deterministic vs LLM-driven orchestration
- Human approval loops

Hands-on:

- Setup and configure Langraph
- Build a multi-agent workflow using Langgraph
- Expose the Langgraph as Fast API service

Outcome:

- Participants can safely orchestrate multi agent complex AI workflows

---

Module 12: Agentic Frameworks & Enterprise Readiness

- Agentic frameworks overview
- CrewAI: role-based agents
- AutoGen: conversational agent coordination
- Agno and emerging frameworks
- Applying MCP, guardrails, and observability to multi-agent systems
- Framework selection and enterprise adoption strategy
- Best practices and guidelines

Hands-on:

- Implement a use case using an alternative agentic framework (CrewAI / Agno / Autogen) and observe multi-agent behavior

Outcome:

- Participants can evaluate, select, and deploy agentic frameworks responsibly in enterprise environments

---