COURSE CURRICULUM

# Gen AI Developer Program

| COURSE DURATION 3 MONTHS | SESSION HOURS 200 HRS | CASE STUDIES & PROJECTS |
| --- | --- | --- |

## Foundations of AI & Machine Learning

### 1. AI & ML Overview

⇨ Definitions: AI vs. ML vs. Deep Learning vs. Generative AI
⇨ Key enterprise use cases & emerging roles
   (AI Architect, Generative AI Developer)

### 2. ML Workflow & Best Practices

⇨ Data ingestion, cleaning, feature engineering
⇨ Training & validation (accuracy, precision, recall, F1)

### 3. Python Data Stack

⇨ NumPy, Pandas, Matplotlib/Seaborn for EDA
⇨ Git & GitHub for version control

### Hands-on Lab

⇨ Basic ML Project: A simple classification (e.g., Titanic survival) to
   practice data workflow, evaluation metrics, and code versioning.

## Deep Learning & Neural Networks

### 1. Neural Network Essentials

⇨ Perceptron, forward & backward propagation, activation functions (ReLU, Sigmoid)

### 2. Hands-On Frameworks

⇨ PyTorch or TensorFlow basics: tensor operations, training loops, model definition

### 3. CNNs & RNNs (Optional Basics)

⇨ CNNs for image tasks (MNIST/CIFAR-10)
⇨ RNNs/LSTMs for sequence data (time permitting)

GEN-AI

## Hands-on Lab

⇨ Train a CNN on MNIST or CIFAR-10 in PyTorch or TensorFlow.

**Performance & Scalability:**

⇨ Performance & Scalability

## Transformer & Large Language Models (LLMs)

### 1.Transformer Architecture

⇨ Self-Attention, Multi-Head Attention, Positional Embeddings

⇨ Encoder-Decoder vs. Decoder-only (GPT-like)

### 2.LLM Ecosystem

⇨ OpenAI GPT (3.5, 4), Google Gemini, Meta LLaMA

⇨ Tokenization (BPE, WordPiece), embeddings, prompt engineering

### 3.Fine-Tuning vs. Prompt Engineering

⇨ **Fine-tuning techniques:** LoRA, Adapters, parameter-efficient methods

⇨ Crafting effective prompts (few-shot, chain-of-thought, system prompts)

### Hands-on Lab

⇨ Prompt Engineering with OpenAI GPT or local LLaMA for text generation.

⇨ Intro to Fine-Tuning: Optional short demonstration of a parameter-efficient fine-tuning method.

### New Emphasis

⇨ Experience in fine tuning open-source models: lay foundational concepts for advanced tuning in subsequent weeks.

### New Emphasis

⇨ Experience in fine tuning open-source models: lay foundational concepts for advanced tuning in subsequent weeks.

GEN-AI

## Retrieval-Augmented Generation (RAG)

### RAG Fundamentals

⇨ Motivation: factual grounding to reduce hallucinations

⇨ Typical pipeline: document chunking, indexing, retrieval, generation

### Vector Databases & Embeddings

⇨ Pinecone, Chroma, Weaviate, Milvus

⇨ Embedding models: OpenAI Embeddings, Sentence Transformers

### Vector Databases & Embeddings

⇨ Pinecone, Chroma, Weaviate, Milvus

⇨ Embedding models: OpenAI Embeddings, Sentence Transformers

### Implementing RAG Pipelines

⇨ Index creation & query mechanism

⇨ Simple Q&A or chat pipeline (LangChain or similar)

### Hands-on Lab

⇨ **RAG Q&A:** Index a small dataset (FAQ, policy docs) in Pinecone or Chroma.

⇨ Build a basic Q&A system retrieving relevant chunks + generating final answers.

### New Emphasis

⇨ Discuss benchmarking retrieval results (precision/recall for domain queries).

⇨ Introduce the concept of domain-specific embeddings for reliability & performance.

## Agentic AI Workflows

### What is an AI Agent?

⇨ Autonomy, multi-step reasoning, planning

### Agent Frameworks

⇨ LangChain Agents, Semantic Kernel, or Crew AI (if relevant)

### Tool & API Orchestration

⇨ Setting up "tools" (e.g., weather API, DB queries, calculator)

⇨ Chain-of-thought prompting & multi-step decision-making

## Hands-on Lab

⇨ Agentic AI Workflow: Build a simple AI Agent that calls a mock external API (e.g., weather or stock data).

⇨ Showcase multi-step reasoning and skill-based agent design.

## New Emphasis

⇨ Innovative AI Applications: Illustrate how to develop agentic workflows for business or industrial contexts (process automation, specialized "skills").

## Fine-Tuning & Benchmarking Domain-Specific Generative Models

### 1. Advanced Fine-Tuning Techniques

⇨ LoRA, Adapters, QLoRA, or full fine-tuning for large open-source models (LLaMA, GPT-Neo)

⇨ Domain adaptation: collecting domain-specific datasets

### 2. Bench marking & Performance Improvement

⇨ Reliability metrics for generative tasks: BLEU, ROUGE, perplexity for text; FID for images

⇨ Scalability considerations (distributed training, multi-GPU setups)

### 3. Bench marking & Performance Improvement

⇨ Reliability metrics for generative tasks: BLEU, ROUGE, perplexity for text; FID for images

⇨ Scalability considerations (distributed training, multi-GPU setups)

### 4. Bias & Robustness

⇨ Reliability metrics for generative tasks: BLEU, ROUGE, perplexity for text; FID for images

⇨ Scalability considerations (distributed training, multi-GPU setups)

## Hands-on Lab

⇨ Domain-Specific Fine-Tuning: Fine-tune an open-source LLM on a small domain dataset, measure improvements vs. baseline.

⇨ Evaluate throughput, latency, memory usage for scalability.

## New Emphasis

⇨ irectly addresses "Fine-tune and benchmark Generative AI models using domain-specific datasets, focusing on performance improvement, reliability, and scalability

## MLOps, AIOps & Production Deployment

### 1.Cloud Platforms & AIOps Pipelines

⇨ Automated deployment & CI/CD with AIOps tools (e.g., Jenkins, GitHub Actions, Tekton)

### 2.Containerization & Serving

⇨ Docker basics, model serving patterns (Kubernetes vs. serverless)

⇨ REST/GraphQL APIs around your generative AI model (FastAPI/Flask)

### 3.Observability & Monitoring

⇨ Logging, metrics, error handling, drift detection

⇨ Setting up monitoring dashboards to track model performance & reliability

### Hands-on Lab

⇨ AIOps Pipeline: Containerize your fine-tuned RAG or AI Agent application and deploy to a chosen platform (AWS ECS, GCP Cloud Run).

⇨ Integrate logs & monitoring for real-time performance tracking.

### New Emphasis

⇨ Explicit coverage of AIOps for multi-environment deployment.

⇨ Revisit Agentic AI to show how skill-based agents can be deployed in production pipelines

## Responsible AI, Real-World Use Cases & Wrap-Up

### 1.Responsible AI Principles
⇨ Bias, fairness, transparency in generative AI
⇨ Prompt security, data privacy, compliance (GDPR, HIPAA)
⇨ Testing solutions against Responsible AI guidelines

### 2.Review & Synthesis
⇨ Recap from ML basics to advanced RAG & AI Agents
⇨ Summaries of fine-tuning and large-scale MLOps

### 3.Career Prep
⇨ Resume/LinkedIn updates emphasizing AI experience
⇨ Interview readiness (ML theory, system design for LLM-driven apps)
⇨ Portfolio building (GitHub, Hugging Face Spaces)

### Hands-on Lab
⇨ **Responsible AI:** Evaluate your fine-tuned model or agentic
   pipeline for bias and policy compliance.
⇨ Conduct a final "mini demo" highlighting best practices.

### New Emphasis
⇨ Reinforces "Knowledge of Responsible AI principles and ways to implement
   it in solution, testing the solution against the defined principles."
⇨ Ties responsible AI checks into final project readiness

## Capstone Workshop

After the 8-week program, participants engage in a Capstone Workshop focusing on end-to-end solution development:

### 1.Project Scoping
⇨ Use domain-specific datasets (text, images, logs, etc.)
   relevant to a chosen business or industrial scenario.
⇨ Define performance, reliability, and scalability goals.

### 2.Data Prep & Model Fine-Tuning
⇨ Apply advanced tuning techniques (LoRA, QLoRA, etc.) on an open-source LLM.
⇨ Benchmark the model with domain metrics (e.g., BLEU, ROUGE, FID, perplexity).

## Python Foundations

### 1.Python Basics

⇨ Data types (strings, lists, tuples, dicts), control flow, functions
⇨ OOP concepts (optional overview)

### 2.Environment Setup

⇨ Virtual environments (venv, conda)
⇨ Basic Git & GitHub for version control

### 3. Data Handling

⇨ File I/O (CSV, JSON)
⇨ Simple reading/writing with Pandas

### Hands-On Lab

⇨ Lab 1: Build a Python script to parse a CSV and produce basic summary stats.
⇨ Push code to GitHub.
**Outcome:** Students establish a solid Python base, environment management, and Git.

## Python Data Structures & ML Basics

### 1. Advanced Python Data Structures

⇨ List/dict comprehensions, sets, decorators (optional)

### 2.Intro to Machine Learning

⇨ Overview of scikit- learn
⇨ Train/test splitting, simple linear/logistic regression

### 3. Evaluation Metrics

⇨ Accuracy, precision, recall, F1

### Hands-On Lab

⇨ Lab 2: Build a simple scikit-learn classifier (e.g., Iris dataset).
⇨ Evaluate model metrics, visualize confusion matrix.
**Outcome:** Comfortable with Python data manipulation and basic ML pipeline.

## FastAPI Essentials

### 1. FastAPI Introduction
⇨ RESTful API concept, asynchronous I/O
⇨ Core features (routers, Pydantic models)

### 2. Basic CRUD
⇨ Handling GET/POST/PUT/DELETE
⇨ Path & query parameters

### 3. API Security
⇨ Oauth2 basics, JWT tokens (high-level overview)

### Hands-On Lab
⇨ **Lab 3:** Create a basic FastAPI app with CRUD endpoints for a mock resource (e.g., "employees").
⇨ Test endpoints using Postman or curl.
**Outcome:** Ability to scaffold a REST API in Python using FastAPI.

## FastAPI Deep Dive & Unit Testing

### 1. Pydantic Models & Validation
⇨ Request/response schemas, data validation

### 2. Unit Testing
⇨ Pytest basics, mocking & fixtures
⇨ Testing FastAPI endpoints

### 3. Deployment Packaging
⇨ Requirements.txt or Pipfile
⇨ Dockerfile basics (intro only)

### Hands-On Lab
⇨ **Lab 4:** Extend the Fast API app with validated data models and write pytest unit tests.
⇨ Optionally create a Docker file for local container testing.
**Outcome:** Students can build a tested, container-ready API with validated schemas

PYTHON

## Deploying Python APIs on AWS

### 1.AWS Overview

⇨ **Key services:** AWS ECS, AWS Elastic Beanstalk, AWS Lambda + API Gateway
⇨ IAM basics (roles, policies)

### 2. Docker & AWS Deployment

⇨ Building Docker images
⇨ Pushing to AWS ECR (Elastic Container Registry)
⇨ Running containers on ECS or Elastic Beanstalk

### 3. Serverless (Optional)

⇨ Brief mention of AWS Lambda

### Hands-On Lab

⇨ **Lab 5:** Containerize the FastAPI application and deploy to AWS ECS.
⇨ Validate endpoints via public URL.
**Outcome:** Real-world exposure to container-based deployment on AWS.

## Deploying Python APIs on Azure

### 1. Azure Basics

⇨ Resource groups, Azure Container Registry (ACR)
⇨ Azure Web App for Containers or Azure Container Instances (ACI)

### 2.Deployment Pipeline

⇨ Docker image push/pull from ACR
⇨ Setting environment variables & app settings

### 3. Monitoring & Logging

⇨ Azure Monitor, App Insights for logs and metrics

### Hands-On Lab

⇨ **Lab 6:** Deploy the same Dockerized FastAPI app to Azure Web App for Containers.
⇨ Review logs and basic metrics in Azure portal.
**Outcome:** Students learn how to replicate a container deployment process on Azure.

PYTHON

## Deploying Python APIs on GCP

### 1. GCP Overview
⇨ Services: Cloud Run, GKE, App Engine

### 2. Cloud Run Deployment
⇨ Containerizing (Docker)
⇨ Submitting images to Google Container Registry
⇨ Configuring environment variables and concurrency

### 3. CI/CD
⇨ GitHub Actions or GCP Cloud Build integration

### Hands-On Lab
⇨ **Lab 7:** Deploy the same FastAPI container to GCP Cloud Run.
⇨ Validate auto-scaling behavior by sending multiple requests.
⇨ **Outcome:** Students see how to deploy container-based Python services on GCP.

## Building a Simple ML Model

### 1. GCP Overview
⇨ scikit-learn pipeline (classification/regression)
⇨ Basic hyperparameter tuning (grid/random search)

### 2. Saving & Loading Models
⇨ Joblib/pickle or ONNX format
⇨ Best practices for reproducibility

### 3. Integration with FastAPI
⇨ Exposing a model inference endpoint
⇨ Handling model load in memory or on demand

### Hands-On Lab
⇨ Lab 8: Create a simple ML model
   (e.g., sentiment classifier or regression model) using scikit-learn.
⇨ Integrate with FastAPI: A /predict endpoint that loads the model and returns predictions.
**Outcome:** Demonstrate how to build a small ML model and wrap it in an API for inference.

## Deploying the ML Model on AWS, Azure & GCP

### 1. Model Serving Approaches

⇨ Containerizing an ML model + FastAPI into one container
⇨ Dealing with environment & memory constraints (esp. for large models)

### 2. Cloud-Specific Details

⇨ AWS: ECS or SageMaker (intro only)
⇨ Azure: Web App for Containers + model storage
⇨ GCP: Cloud Run or Vertex AI (intro only)

### 3. Best Practices

⇨ Logging predictions, monitoring model performance in each cloud
⇨ Cost management & scaling concerns

### Hands-On Lab

⇨ **Lab 9:** Deploy the ML + FastAPI container to each cloud (or pick your favorite).
⇨ Validate that your inference endpoint is accessible, stable, and logs performance metrics.
**Outcome:** Students learn to host a simple ML model on AWS, Azure, and GCP,
reinforcing multi-cloud concepts with a working example.

### Hands-On Lab

⇨ **Lab 10:** Final Project: A minimal end-to-end pipeline from local dev
⇨ Docker container
⇨ ML model
⇨ cloud deployment.
**Outcome:** A portfolio-ready final project showcasing a containerized ML-powered API deployed on at least one major cloud provider.

## Summary of Curriculum

⇨ **Python & FastAPI:** Students learn Python basics and build robust REST APIs.

⇨ **Testing & Containerization:** Master unit testing (pytest) and Docker packaging.

⇨ **Cloud Deployments:** Deploy the same containerized application to AWS, Azure, and GCP.

⇨ **Building & Hosting an ML Model:** Integrate a simple scikit-learn model into FastAPI and deploy to the cloud.

⇨ Final Capstone: A fully tested, containerized ML solution running in production-like cloud environments.


By the end, participants will be confident in Python fundamentals, API development, and deploying ML-driven apps across multiple cloud platforms—an essential skill set for developers moving toward Generative AI and more advanced ML solutions.

```
pt>
ion show(shown, hidden) {
ument.getElementById(shown).style.displ
ument.getElementById(hidden).style.di
rn false;
        1  <html>
        2  <head>
        3  <script>
        4  function show(shown, hidden) {
        5     document.getElementById(shown).style.display
        6     document.getElementById(hidden).style.displa
        7     return false;
        8  }
        9  </script>
       10  </head>
       11  <body>
       12
       13  <div id="Page1">
       14    Content of page 1
       15    <a href="#" onclick="return sh
```

## Our Students Are Placed In