DATA SCIENCE with Generative AI & Agentic Systems

From Python Foundations to Production-Grade AI Systems

Vision of the Institute:

To empower the next generation of AI professionals by delivering industry-focused education that bridges the gap between academic learning and real-world applications. Our mission is to develop highly skilled AI engineers capable of building intelligent, scalable, and production-ready solutions for modern enterprises.

Program Overview:

Artificial Intelligence is transforming every major industry, including Finance, Banking, Healthcare, E-Commerce, Manufacturing, Real Estate, Cybersecurity, SaaS, and global digital platforms.

This comprehensive **9-phase career-oriented program** is designed to take you from **Python programming fundamentals** to **production-ready AI systems**. You'll gain hands-on experience in Data Science, Machine Learning, Deep Learning, Natural Language Processing, Large Language Models (LLMs), Generative AI, Retrieval-Augmented Generation (RAG), Agentic AI Systems, and MLOps.

The program concludes with an **industry-grade capstone project**, where you'll build, deploy, and showcase a complete AI application using modern technologies such as **React, FastAPI, Docker, and AWS Cloud**.

Who This Program Is For:

- 2nd, 3rd, and Final-Year Engineering Students
- Fresh Graduates
- Anyone passionate about building a career in Artificial Intelligence

Technical Prerequisites:

- No prior AI experience required. Curriculum begins with Python fundamentals and progresses step-by-step to advanced AI systems
- Logical thinking, and willingness to learn are essential.

Learning Outcomes: By the end of this program, you will be able to:

- Build **25+ hands-on mini projects** across Data Analysis, Machine Learning, Deep Learning, NLP, Generative AI, and Agentic AI.
- Develop an **industry-grade end-to-end capstone project** from problem identification to deployment.

Skills Gained — At a Glance:

Programming & Software Engineering	**Data Science & Machine Learning**	**Generative AI & Production AI**
Python Programming	NumPy & Pandas	Prompt Engineering
Data Structures & Algorithms	Data Cleaning & Wrangling	Large Language Models (LLMs)
Object-Oriented Programming (OOP)	Data Visualization	Transformers (BERT, GPT, etc.)
Git & GitHub	SQL (MySQL)	AI Application Development & Deployment
Linux Fundamentals	MongoDB	Retrieval-Augmented Generation (RAG)
REST APIs	Power BI Dashboards	Vector Databases (FAISS, ChromaDB, Pinecone)
FastAPI	Machine Learning	Semantic Search & Hybrid Search
React	Feature Engineering	LangChain, LangGraph
Docker	Model Evaluation & Hyperparameter Tuning	CrewAI & Agent Orchestration
CI/CD (GitHub Actions)	Deep Learning (PyTorch)	AI Agents & Multi-Agent Systems
Cloud Deployment (AWS)	Computer Vision (CNNs)	Model Context Protocol (MCP)
System Design Fundamentals	Natural Language Processing (NLP)	Agent-to-Agent (A2A) Communication
API Development & Integration	Time Series & Recommendation Systems	LLM Evaluation & Testing
Testing & Debugging	Model Deployment	Tool Calling & Function Calling
Monitoring & Observability	MLOps & Model Monitoring	AI Guardrails & Safety

PHASE 01

Python Programming Foundations

Core Syntax · OOP · Data Structures · Algorithms · File Handling

Core Syntax & Logic

- Python syntax & execution flow
- Data types, variables & operators
- Conditional statements & control flow
- Loops, patterns & nested logic
- String manipulation
- List comprehensions

Data Structures & Algorithms

- Lists, tuples, sets & dictionaries
- Searching & sorting algorithms
- Stack & Queue implementations
- Time complexity (Big-O notation)

Advanced Python

- Functions & recursion
- Lambda, map, filter & reduce
- Decorators & generators
- Regular expressions
- Modules & Packages
- Logging
- OOP — classes, objects, inheritance, Encapsulation, Polymorphism, Abstraction
- Exception handling
- File & I/O handling
- Type Hints
- Dataclasses
- Pydantic
- Os, requests library
- Environment variables (python-dotenv)
- Concurrency
- Working with dates/times & JSON
- Virtual Environments & Dependency Management (pip, uv)
- Testing with pytest

What You Will Build

- A command-line task manager using Python OOP principles and file I/O
- A text-based data processing utility using string manipulation and list comprehensions
- Mini Python package with modules, virtual environment, and pip-installable structure

OUTCOME: *Strong Python coding foundation for ML & AI backend systems — interview-ready and production-capable*

PHASE 02

Data Analysis & Visualization

NumPy · Pandas · Matplotlib · Seaborn · Plotly · SQL · MongoDB · Power BI

Numerical Computing

- NumPy arrays & broadcasting
- Array indexing, slicing & boolean masking
- Vectorized operations & reshaping
- Linear algebra basics
- Random number generation

Data Wrangling — Pandas

- DataFrames & Series
- Reading & writing data — CSV, Excel, JSON, SQL
- Data cleaning & imputation
- Feature engineering
- Merging & reshaping data
- GroupBy & aggregations
- Datetime data handling

Visualization & Databases

- Matplotlib charts & subplots
- Seaborn statistical plots
- Plotly interactive charts
- EDA — Univariate, Bivariate & Multivariate analysis, Correlation analysis, Outlier detection, Data quality assessment and Business insight generation
- SQL — joins, subqueries, window functions

- Connecting Python to databases (SQLAlchemy)
- MongoDB — CRUD & aggregation pipelines

Power BI & Dashboarding

- Power BI — reports & dashboards
- DAX measures & calculated columns
- Data storytelling & KPI design principles
- Google Looker Studio

What You Will Build

- End-to-end exploratory data analysis (EDA) notebook on a real-world dataset
- Interactive Plotly dashboard presenting key business metrics and trends
- MongoDB-backed Python application using CRUD operations and aggregation pipelines
- Power BI report with DAX measures, KPI cards, and data storytelling narrative

OUTCOME: *Industry-ready data handling, wrangling, and visual storytelling skills*

PHASE 03

Mathematics for Data Science

Probability · Inferential Stats · Hypothesis Testing · Linear Algebra · Calculus

Probability & Distributions

- Probability theory fundamentals
- Bayes' theorem
- Normal & binomial distributions
- Poisson distribution
- Sampling methods
- Central limit theorem

Descriptive & Inferential Statistics

- Mean, median, mode & spread
- Variance & standard deviation
- Confidence intervals
- Hypothesis testing (t-test, z-test)
- Chi-square & ANOVA

- p-values & A/B testing

Linear Algebra & Calculus

- Vectors & matrix operations
- Eigenvalues & eigenvectors
- Derivatives & chain rule
- Gradient descent intuition
- Partial derivatives

OUTCOME: *Mathematical confidence to understand and build ML & DL models from scratch*

PHASE 04

Machine Learning — Core Algorithms

Scikit-learn · Regression · Trees · SVM · Clustering · Evaluation

Supervised Learning

- Linear & Logistic Regression
- Ridge & Lasso (Regularization)
- K-Nearest Neighbors (KNN)
- Decision Trees & Random Forests
- Gradient Boosting (XGBoost)
- LightGBM
- CatBoost
- Support Vector Machines (SVM)
- Naive Bayes classifier
- Ensemble methods

Unsupervised Learning

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN (density-based)
- PCA — Dimensionality Reduction
- t-SNE visualization
- Anomaly Detection methods

Model Engineering

- Bias-Variance tradeoff
- Cross-validation strategies
- Feature selection & importance
- Hyperparameter tuning (GridSearch)
- Evaluation metrics
- Model interpretability (SHAP, LIME)
- Imbalanced datasets (SMOTE)
- Pipeline building
- Saving & loading models
- Hands-on ML Mini Projects

What You Will Build

- End-to-end pipeline: data prep → feature engineering → XGBoost → evaluation report
- Customer segmentation system using K-Means and PCA with interactive Plotly visualizations
- Anomaly detection engine for financial transactions using DBSCAN and Isolation Forest
- Scikit-learn ML pipeline with GridSearch tuning, cross-validation and model serialization

OUTCOME: *Confidence to build, evaluate, and explain any ML model — in interviews and in production*

PHASE 05

Deep Learning & Computer Vision

TensorFlow/PyTorch · CNNs · Transfer Learning · Object Detection ·

Neural Network Foundations

- Neurons & network architecture design
- Activation & loss functions
- Optimizers — Adam, SGD, RMSProp
- Backpropagation & gradient flow
- Overfitting, dropout & regularization
- Batch normalization

Computer Vision

- CNNs — Conv, Pool, Flatten layers
- Transfer learning (ResNet, EfficientNet)
- Object detection with YOLO

What You Will Build

- Custom CNN trained from scratch on a multi-class image classification dataset
- Transfer learning pipeline using ResNet/EfficientNet for domain-specific image recognition
- Real-time object detection system powered by YOLO on live video or image input

OUTCOME: *Build real-world image-based AI systems — using state-of-the-art architectures*

PHASE 06

NLP, Transformers & Large Language Models

GPT · HuggingFace · Sentiment . Prompt Engineering

NLP Fundamentals

- Text preprocessing & tokenization
- Bag-of-Words & TF-IDF
- Word2Vec embeddings
- Sentiment analysis
- Named Entity Recognition (NER)
- Text classification

Sequence Models

- RNN, LSTM, GRU architectures
- Bidirectional RNNs
- Encoder-Decoder architecture
- Attention mechanism deep dive

Transformers & LLMs

- Transformer architecture — Self-attention, Multi-head attention, Positional encoding, Encoder, Decoder, Residual connections, Layer normalization, Feed-forward networks, Masking, Tokenization, Context windows
- BERT
- GPT series — architecture & usage

- HuggingFace pipelines & models
- Prompt Engineering strategies
- Embeddings & vector search
- Gemini / Claude / GPT API integration
- Chatbot development with LLMs

What You Will Build

- Sentiment Analysis using TF-IDF, Word Embeddings, and LSTM
- AI Chatbot using GPT/Gemini/Claude APIs with Prompt Engineering
- Semantic Search Application using Embeddings and Vector Search

OUTCOME: *Master the fundamentals of Transformers and Large Language Models to build production-ready NLP applications, semantic search systems, and AI-powered chatbots.*

PHASE 07

Generative AI, RAG & Agentic Systems

LangChain · LangGraph · Agents · Vector DBs · MCP · A2A

LLM Application Foundations

- LangChain fundamentals
- Prompt Templates
- Chains & LCEL (LangChain Expression Language)
- Output Parsers & Structured Outputs
- Streaming
- LangSmith (tracing & debugging)
- Model selection strategies
- Token counting & cost optimization

Context Engineering

- Data ingestion & preprocessing
- Document chunking strategies
- Metadata enrichment
- Embeddings & semantic search
- Vector databases (FAISS, Chroma, Pinecone)
- Context window management
- LLM application architecture

Retrieval-Augmented Generation (RAG)

- RAG system design & architecture
- Retrieval techniques & hybrid search strategies
- Query understanding & query rewriting
- Prompt engineering for RAG
- Memory & conversational RAG
- Self-RAG & Graph RAG
- Agentic RAG
- RAG evaluation metrics

Agentic Systems — Core Concepts

- Foundations of Agentic AI
- Core Agent Architecture
- Tool Use / Function Calling — external APIs, Python tools, database tools, web search tools, calculator tools, file system tools
- Planning & Reasoning
- Memory Systems
- Multi-Agent Systems

Orchestration Frameworks

- LangGraph — graph architecture, state management, nodes, edges, conditional routing, cycles, checkpointing
- LangChain Agents
- CrewAI

Agent Communication Protocols & Standards

- MCP (Model Context Protocol)
- A2A (Agent-to-Agent protocol)
- Function/tool schema standards
- Agent discovery mechanisms

Autonomous Agent Design Patterns

- Supervisor pattern
- Planner–Executor pattern
- Hierarchical agents
- ReAct
- Tree of Thoughts
- Reflection & self-consistency
- Planning loops
- Deep Research agents

- Human-in-the-Loop
- Agent streaming

Evaluation, Safety & Observability

- Prompt evaluation
- Agent evaluation
- RAG evaluation
- Tracing, logging & monitoring
- AI Guardrails & Safety
- Responsible AI

What You Will Build

- Production RAG system: PDF ingestion → ChromaDB/FAISS → semantic retrieval → grounded LLM response
- Autonomous research agent using LangChain + tool calling (web search, calculator, database lookup)
- Multi-agent orchestration system using LangGraph with stateful memory and task delegation between agents
- CrewAI orchestration system: role-based automated content pipeline with agent personas (Researcher, Writer, Editor)
- Human-in-the-Loop (HITL) workflow: high-stakes execution agent built with LangGraph
- Deep Agent planner: complex reasoning agent with Tree of Thoughts / ReAct for multi-step research
- Model Context Protocol (MCP) integration: custom MCP server bridging local environment with an LLM

OUTCOME: *Design and build autonomous AI systems powering modern real-world AI products*

PHASE 08

MLOps & Model Deployment

FastAPI · React JS. Docker · Git · Cloud · CI/CD · Monitoring

Version Control & Tracking

- Git & GitHub
- Branching & pull requests
- Model registry & versioning

Model Serving & APIs

- FastAPI endpoint design
- REST API design principles
- API documentation with Swagger
- Authentication & security basics (JWT)

Frontend for AI Apps

- JavaScript fundamentals
- Linux Fundamentals
- React basics for AI-powered UIs

Containerisation & CI/CD

- Docker fundamentals
- Dockerizing ML applications
- GitHub Actions (CI/CD pipelines)

Cloud Deployment

- AWS (EC2, S3, RDS)
- Cloud deployment strategies

Monitoring & Observability

- Application & model logging
- Model performance monitoring & drift detection
- Basic observability tooling (Prometheus/Grafana concepts)

What You Will Build

- FastAPI-served ML model with JWT authentication, versioned endpoints, and Swagger documentation
- Dockerised ML application with Docker Compose for multi-service orchestration

- GitHub Actions CI/CD pipeline: automated testing → build → deploy on every commit

OUTCOME: *Deploy and maintain production-grade AI models like a professional ML Engineer*

PHASE 09

Capstone Project — Industry Grade

Data → Model → API → Dashboard → Cloud → Portfolio

Project Execution

- Real-world problem statement selection
- Data collection & pipeline design
- EDA & feature engineering
- Model building & evaluation
- Generative AI or Agentic component
- FastAPI backend development
- Docker containerisation
- Deployment to cloud platform

Career Preparation

- GitHub portfolio setup
- Resume preparation guidance
- Technical interview preparation
- Mock interviews with feedback

Capstone Deliverables

- A GitHub repository with clean code, README, architecture diagram, and demo instructions
- A deployed web application or API accessible via a public URL

OUTCOME: *A job-winning, fully deployed AI portfolio project ready for top industry roles*

Tools & Technologies





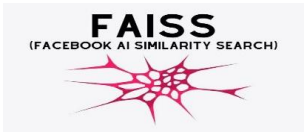
















































Deep Agents	Pydantic	FastAPI
git	GitHub	GitHub Actions
docker	GitHub Copilot	Claude
aws	React	