KRISHNAIK ACADEMY

# ULTIMATE RAG

V 1.0

RETRIEVAL AGENTIC MULTIMODAL

DEPLOYMENT ORCHESTRATION

Designed for AI engineers, ML practitioners, and software developers, this course transforms RAG theory into production-grade implementations. Whether you're new to retrieval systems or experienced with LLMs, you'll gain expertise in Building scalable RAG pipelines (from basic similarity search to knowledge graph-enhanced retrieval), Implementing enterprise features (hybrid search, MCP integration, self-improving agents), Deploying optimized systems (Docker, FastAPI, Kubernetes, AWS EKS), Solving real-world challenges (multimodal processing, conversational memory, code-aware RAG). Through hands-on projects, you'll create portfolio-worthy systems including MCP-integrated code assistants and cloud-deployed multimodal RAG, all with professional monitoring and evaluation frameworks.

## Learning Objectives

✅ Implement End-to-End RAG: From document ingestion (Unstructured.io, LlamaParse) to response synthesis (GPT-5, Claude 3.5)
✅ Self-RAG with confidence scoring
✅ Graph RAG with entity resolution
✅ Model Context Protocol (MCP) integration
✅ Optimize Production Systems:
✅ Kubernetes orchestration (EKS)
✅ Advanced caching/load balancing
✅ CI/CD with GitHub Actions/AWS CodePipeline
✅ Build Specialized Applications:
✅ Multimodal RAG (CLIP + Whisper)
✅ Smart code review with MCP servers
✅ Agentic workflows with ReAct

# Course Information

## Prerequisites

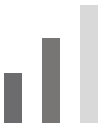
A  solid foundation in Python programming, including familiarity with async/await patterns and type hints. Learners should understand basic machine learning concepts like embeddings and tokenization, with prior exposure to REST APIs or cloud services (AWS/Azure) being advantageous but not mandatory. While the curriculum covers everything from foundational RAG to advanced agentic systems, those with experience in NLP frameworks (LangChain/LlamaIndex) or containerization (Docker) will progress faster through the production-focused modules.

Over 4 months, this intensive program guides you from RAG fundamentals to cutting-edge implementations like Model Context Protocol (MCP) integration and multimodal systems. Through 16 meticulously structured modules and five hands-on projects, you'll master both theory and practice building everything from basic Q&A pipelines to cloud-deployed, Kubernetes-orchestrated RAG applications. The course emphasizes real-world viability, teaching CI/CD workflows, hybrid search optimization, and professional evaluation techniques while leveraging the latest tools (GPT-5, Claude 3.5, Weaviate) and paradigms like self-improving agents and graph-enhanced retrieval. By the end, you'll be equipped to design, deploy, and maintain enterprise-grade RAG solutions.

## Estimated Time

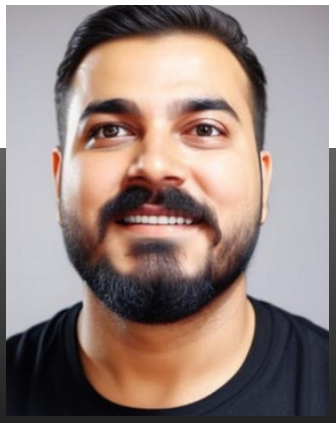
4 months 6hrs/week*

## Required Skill Level



Intermediate

# Course Instructors



**Krish Naik**
*Chief AI Engineer*
Linkedin

**Sourangshu Pal**
*Senior Data Scientist*
Linkedin

**Sunny Savita**
*GenAI Engineer*
Linkedin

**Mayank Aggrawal**
*Senior ML Engineer*
Linkedin

Module 1

# Foundations of RAG

This module establishes the core concepts of Retrieval-Augmented Generation, starting with its fundamental principles and advantages over traditional LLM approaches. You'll explore the complete RAG pipeline from document processing to response generation, while setting up a practical development environment with essential tools like LangChain and vector databases. By the end, you'll understand how to evaluate trade-offs between different RAG implementations and configure basic systems efficiently.

| Topics | |
|---|---|
| **Introduction to RAG** | What is RAG and why it matters, Limitations of pure LLMs, RAG vs fine-tuning vs prompt engineering, Real-world applications and use cases |
| **Core Components of RAG** | Document ingestion and preprocessing, Embedding models and vector representations, Vector databases fundamentals, Retrieval mechanisms, Generation and response synthesis |
| **Setting Up Your Development Environment** | Python environment setup, Installing necessary libraries (LangChain, LlamaIndex, OpenAI, etc.), Vector database options (Pinecone, Weaviate, ChromaDB), Cost considerations and API management |

Module 2

# LangChain

This module introduces LangChain's core architecture for building RAG applications. You'll learn document ingestion methods, various text splitting techniques for optimal chunking, and how to work with different embedding models. The module also covers vector store implementations for efficient retrieval.

| Topics | |
|---|---|
| **LangChain Fundamentals** | Introduction to basic components and modules in LangChain, Data ingestion with documents loaders |
| **Text Processing** | Text splitting techniques-Recursive Character Text Splitter, Text splitting technique-Character Text Splitter, Text splitting technique-HTML Header Text Splitter, Text splitting technique-Recursive Json Splitter |
| **Embeddings** | Introduction to OpenAI Embedding, Ollama Embeddings, Huggingface Embeddings |
| **Vector Stores** | VectorStores-FAISS, VectorStore and Retriever-ChromaDB |

Module 3

# Traditional RAG Implementation

This module guides you through implementing a complete traditional RAG system, from setting up basic similarity search to building a functional Q&A application. You'll learn prompt engineering for retrieval tasks, response formatting techniques, and comprehensive evaluation methods to assess and optimize your RAG pipeline.

| Topics | |
|---|---|
| **Basic RAG Pipeline** | Basic similarity search |
| **Building RAG Applications** | Implementing a simple Q&A system, Prompt engineering for RAG, Handling retrieval results, Response generation and formatting |
| **Evaluation and Testing** | RAG evaluation metrics (relevance, faithfulness, answer quality), Building test datasets, A/B testing different configurations, Performance optimization |

Module 4

# Working with Document Parsers

This module introduces specialized document parsing tools for RAG systems, covering their installation, configuration, and integration into document processing pipelines.

| Topics | |
|---|---|
| **Document Parsing Tools** | Doclens, LLama Parse, unstructured, vectorize |

Module 5

# LlamaIndex

This module covers LlamaIndex from setup to implementation, teaching you how to create, query, and optimize vector indexes for efficient document retrieval.

| Topics | |
|---|---|
| **LlamaIndex Fundamentals** | Introduction to LlamaIndex: What it is and why it matters, Installing and configuring LlamaIndex in Python, Understanding core concepts: Index, Document, Query Engine |
| **Document Handling** | Loading documents from various sources (PDF, Notion, web, local) |
| **Indexing and Querying** | Creating a VectorStoreIndex using OpenAI or SentenceTransformers, Querying the index with natural language questions, Using response synthesizers (Tree, Compact, Refinement) |
| **Hybrid Search & Query Optimization** | Hybrid RAG (keyword + vector + metadata), Query engines with filters, Reranking capabilities |

Module 6

# Advanced LlamaIndex Techniques

This module explores professional-grade LlamaIndex implementations, covering custom configurations, advanced retrieval methods, system integrations, and query optimization techniques for production environments.

| Topics | |
|---|---|
| **Custom Configurations** | Using ServiceContext, StorageContext, and LLMPredictor for custom setups, Chunking strategies: fixed, recursive, and semantic |
| **Advanced Retrieval** | Advanced retrievers, Keyword Table Retriever, Vector Index Retriever, Auto Merging Retriever |
| **System Integration** | Streaming responses with LlamaIndex, Integrating LlamaIndex with LangChain or FastAPI, Adding tool/function calling support using ToolRetriever |
| **Query Optimization** | Hybrid RAG: Combining keyword + vector + metadata-based search, Query engines with filters and reranking capabilities |

Module 7

# Haystack

This module teaches you to implement RAG systems using Haystack, covering pipeline construction, metadata handling, advanced query techniques, and specialized applications including multimodal and agentic implementations.

| Topics | |
|---|---|
| **Haystack Fundamentals** | Introduction to Haystack, Working with Pipelines |
| **Core RAG Implementation** | Building Basic RAG with Haystack, Filtering Documents with Metadata, Improving Retrieval by Embedding Meaningful Metadata |
| **Advanced Techniques** | Using Hypothetical Document Embedding (HyDE), Query Decomposition and Reasoning, Query Expansion, Automated Structured Metadata Enrichment |
| **Specialized Applications** | Classifying Documents & Queries by Language, Building an Agentic RAG with Fallback to Websearch, Creating Vision+Text RAG Pipelines, Evaluating RAG Pipelines |

Module 8

# LangGraph Introduction

This module introduces LangGraph for building stateful LLM workflows, covering setup, graph construction, chatbot development, agent architectures, and production-ready features like streaming and human-in-the-loop workflows.

| Topics | |
|---|---|
| **Setup & Fundamentals** | Introduction To LangGraph, Getting Started LangGraph Application - Creating The Environment, Setting Up OpenAI API Key, Setting Up GROQ API Key, Setting Up LangSmith API Key |
| **Core Graph Concepts** | Developing A Simple Graph or Workflow Using LangGraph - Building Nodes And Edges, Building Simple Graph StateGraph And Graph Compiling, State Schema With DataClasses, Pydantic, Chain In LangGraph, Routers In LangGraph |
| **Chatbot Development** | Developing LLM Powered Simple Chatbot Using LangGraph, Tools And ToolNode With Chain Integration - Part 1, Building Chatbot With Multiple Tools Integration - Part 1, Building Chatbot With Multiple Tools Integration - Part 2 |
| **Advanced Architectures** | Introduction To Agents And ReAct Agent Architecture In LangGraph, ReAct Agent Architecture Implementation, Agent With Memory In LangGraph, Prompt Chaining, Prompt Chaining Implementation With Langgraph, Parallelization, Routing |

| Topics | |
|---|---|
| **Production Features** | Streaming In LangGraph, Streaming using astream events Using Langgraph, Orchestrator-Worker, Orchestrator Worker Implementation, Human In The Loop With LangGraph Workflows |

Module 9

# Enhanced RAG Techniques

This module explores professional-grade RAG optimization techniques, covering advanced document processing, hybrid retrieval methods, and query improvement strategies to significantly enhance system performance.

| Topics | |
|---|---|
| **Advanced Chunking and Preprocessing** | Semantic chunking, Sliding window approaches, Document hierarchy preservation, Metadata extraction and usage |
| **Hybrid Search Strategies** | Combining dense and sparse retrieval, BM25 + semantic search, Re-ranking techniques, MMR (Maximal Marginal Relevance) |
| **Query Enhancement** | Query expansion techniques, Query decomposition, Hypothetical document embeddings (HyDE), Query routing |

Module 10

# Multi-Modal and Structured RAG

This module extends RAG capabilities to structured data and multi-modal content, teaching integration with databases/APIs and techniques for processing mixed-format documents (images, tables, audio/video).

| Topics | |
|---|---|
| **Structured Data RAG** | RAG over SQL databases, Integrating with APIs, JSON and CSV processing, Knowledge graph integration |
| **Multi-Modal RAG** | Image and text combined retrieval, Working with PDFs containing images/tables, Audio and video transcript RAG, Cross-modal search strategies |

Module 11

# Conversational and Contextual RAG

This module enhances RAG systems with conversational memory and personalization capabilities, teaching context management, chat interface development, and user-adaptive retrieval while addressing privacy requirements.

| Topics | |
|---|---|
| **Memory Systems for RAG** | Conversation history management, Context window optimization, Memory summarization techniques, Long-term vs short-term memory |
| **Conversational RAG Implementation** | Chat-based interfaces, Context-aware retrieval, Follow-up question handling, Conversation flow management |
| **Personalization in RAG** | User profiling, Preference-based retrieval, Adaptive response generation, Privacy considerations |

Module 12

# Agentic RAG Fundamentals

This module introduces agentic RAG systems that actively plan and reason, covering core architectures, implementation patterns like ReAct, and multi-step query resolution techniques.

| Topics | |
|---|---|
| **Agentic RAG Concepts** | What makes RAG 'agentic', Agent architectures overview, Tools and function calling, Decision-making in retrieval |
| **Agent Implementation** | ReAct pattern implementation, Tool creation for RAG agents, Chain-of-thought reasoning, Self-reflection mechanisms |
| **Complex Reasoning** | Query planning and decomposition, Iterative retrieval strategies, Answer synthesis from multiple sources |

Module 13

# Advanced Agentic RAG

This module explores cutting-edge agentic RAG implementations, teaching self-optimizing autonomous agents, multi-agent collaboration frameworks, and production-grade orchestration techniques.

| Topics | |
|---|---|
| **Autonomous Agents** | Self-improving retrieval systems, Dynamic tool selection, Adaptive retrieval strategies, Learning from user feedback |
| **Multi-Agent Systems** | Specialized agent roles, Agent communication protocols, Collaborative retrieval and generation, Consensus mechanisms |
| **Production Orchestration** | Workflow automation, Complex query handling, Error recovery and fallback strategies |

Module 14

# Production RAG Systems

This module covers professional RAG deployment, teaching scaling techniques and monitoring practices for maintaining high-performance systems in production.

| Topics | |
|---|---|
| **Scaling Techniques** | Caching strategies, Load balancing, Asynchronous processing |
| **Deployment & Monitoring** | Container deployment, API design for RAG services, Logging and observability, Performance metrics |

Module 15

# Cutting-Edge RAG Techniques

This module explores frontier RAG innovations including self-correcting retrieval systems, knowledge graph integration, and specialized fine-tuning approaches for optimal performance.

| Topics | |
|---|---|
| **Self-RAG & Adaptive Retrieval** | Retrieval confidence scoring, Dynamic retrieval decisions, Self-critique mechanisms, Adaptive chunk sizing |
| **Graph RAG** | Building knowledge graphs from documents, Graph-based retrieval, Entity linking and resolution, Reasoning over graph structures |
| **Fine-Tuning** | Retriever fine-tuning, Reader model optimization, End-to-end RAG training, Domain adaptation |

Module 16

# Model Context Protocol (MCP)

This module covers Model Context Protocol from theory to implementation, including core architecture, existing integrations, and custom server development.

| Topics | |
|---|---|
| **MCP Fundamentals** | Introduction to Model Context Protocol, Important components of MCP, Communication between components of MCP |
| **MCP Implementations** | Demo of MCP with Claude Desktop, Cursor IDE MCP, Claude Desktop with MCP, Claude with MCP, Exploring MCP repositories like Smithery.ai |
| **Advanced Development** | Building MCP servers with tools and client from scratch using LangChain, Docker MCP catalog and toolkit |

Module 17

# End-to-End Projects

Through these five progressive projects, you'll evolve from building basic RAG systems to implementing advanced multimodal and agentic workflows. Starting with containerized Q&A pipelines (Project 1), you'll scale to persistent multi-format retrieval (Project 2), implement hybrid search (Project 3), then tackle multimodal challenges (Project 4). The finale integrates MCP for contextual code analysis (Project 5) - delivering hands-on experience with FastAPI, Docker, AWS, and cutting-edge RAG optimizations.

| Topics | |
|---|---|
| **Project 1 : Basic RAG Q&A System with CI/CD Foundation** | **Stack**: FastAPI, Python 3.12, OpenAI GPT-5, FAISS, LangChain, PyPDF2, Docker, GitHub Actions<br><br>**Key Learnings**: Basic RAG pipeline implementation, Git workflow for ML projects, Simple CI/CD with GitHub Actions, Docker containerization basics, API development with FastAPI<br><br>**Deliverables**: Dockerized FastAPI application, GitHub repository with CI/CD pipeline, Basic test suite, API documentation with Swagger |
| **Project 2: Multi-Source RAG with Vector Database** | **Stack**: FastAPI, Python 3.12, OpenAI GPT-5, ChromaDB, Sentence-Transformers, Unstructured.io, Docker Compose |

| Topics | |
|---|---|
| | **Key Learnings**: Persistent vector storage, Multi-format document processing (PDF/DOCX/CSV/JSON), Docker Compose orchestration, Advanced CI/CD, Basic monitoring implementation<br><br>**Deliverables**: Document ingestion API with multi-format support |
| **Project 3: Hybrid Search RAG with Advanced Chunking** | **Stack**: FastAPI, Python 3.12, Anthropic/OpenAI LLM, Weaviate, Elasticsearch, BGE embeddings, Cohere reranker<br><br>**Key Learnings**: Hybrid search (BM25 + semantic), Advanced chunking strategies, Kubernetes basics, Re-ranking techniques<br><br>**Deliverables**: Hybrid search API with optimized retrieval |
| **Project 4: Multimodal RAG (Text + Images + Audio)** | **Stack:** FastAPI, Python 3.12, GPT-4 Vision/Claude, Milvus, CLIP, Whisper, AWS S3/EKS<br><br>**Key Learnings:** Multimodal embedding strategies, Cross-modal retrieval, AWS cloud integration (S3, EKS), Audio/image processing<br><br>**Deliverables:** Multimodal search interface, AWS EKS deployment |

| Topics | |
|---|---|
| **Project 5: Smart Code Review & Bug Fix Assistant** | **Stack:** FastAPI, LangChain, ChromaDB, MCP Servers (Git/GitHub/Filesystem), OpenAI/Claude 3.5, Docker<br><br>**Key Learnings:** MCP server integration, Code-aware RAG systems, Context-aware debugging, Multi-repository analysis<br><br>**Deliverables:** Code review assistant with MCP context awareness |