Enterprise Agentic RAG: Master Syllabus

Phase 1: Foundations & Structured Logic

Goal: Move from chatting to building systems that exchange data.

Key Topics:

- Async Python: Handling multiple LLM calls simultaneously.
- Pydantic & JSON Schemas: Forcing LLMs to output structured data.
- Token Economics: Context windows, cost optimization, and prompt caching.
- Vector DB Fundamentals: Qdrant/Pinecone basics and Metadata filtering.

Capstone Project: The Metadata-Aware PDF Chatbot

Build a system that takes 20 PDFs, extracts metadata (Title, Author, Date), and allows users to search only within specific time ranges using metadata filters.

Phase 2: Advanced Data Engineering (The 'Raj' Layer)

Goal: Solve the 'Garbage In, Garbage Out' problem.

Key Topics:

- Intelligent Parsing: Mastering Docling or Unstructured.io for complex layouts.
- The Quality Scorer: Detecting blurry scans or bad OCR before ingestion.
- Table Extraction Mastery: Converting multi-page tables into clean Markdown.
- Hierarchical Chunking: Implementing Parent-Child strategies.

Capstone Project: The Table-Heavy Financial Analyst

Build a RAG system that processes a 100-page Annual Report with 20+ tables. The AI must answer specific numerical questions with 100% accuracy.

Phase 3: Agentic Reasoning & Tool Orchestration

Goal: Giving the AI 'hands' to use tools and 'brains' to plan.

Key Topics:

- Function Calling: LLM triggering Python functions (Calculators, SQL, Search).
- The ReAct Pattern: Implementing 'Reason + Act' loops.
- Query Decomposition: Breaking one prompt into multiple search steps.
- Agentic Routing: Deciding between Vector DB, SQL, or Web Search.

Capstone Project: The Multi-Tool Researcher

Build an agent that can answer: 'What was Tesla's revenue in 2023 and how does it compare to the current market cap on Google?' (Vector + Web + Math).

Phase 4: Cognitive Architecture (Context & State)

Goal: Building 'Contextual Awareness' and Memory.

Key Topics:

- State Management (LangGraph): Building cyclic graphs to fix mistakes.
- Memory Systems: Short-term thread history vs. Persistent long-term profiles.
- Contextual Compression: Summarizing history to save context space.
- Multi-Agent Orchestration: One agent 'Searches' and another agent 'Reviews'.

Capstone Project: The Persistent Legal Assistant

Build an agent that remembers legal preferences across sessions and manages a 10-step discovery process without losing track.

Phase 5: Self-Correction & Evaluation

Goal: Eliminating hallucinations and ensuring 'Enterprise Truth'.

Key Topics:

- Corrective RAG (CRAG): Building a 'Grader' that rejects bad search results.
- Self-RAG: An agent that critiques its own output and searches again.
- RAGAS Framework: Using AI to grade Faithfulness and Relevance.
- Hallucination Filters: Fact verification against source text.

Capstone Project: The Zero-Hallucination Pharma Bot

Build a clinical trial system where the agent refuses to answer unless it finds three matching citations in the source text.

Phase 6: Reinforcement Learning & Optimization

Goal: Making the system get smarter every time it is used.

Key Topics:

- Feedback Loops (RLAIF): User Upvote/Downvote UI for training data.
- Reward Modeling: Using a 'Judge LLM' to grade search strategies.
- Contextual Re-ranking: Fine-tuning ranking based on historical success.
- Self-Healing Pipelines: Learning to ignore low-quality document folders.

Capstone Project: The Self-Improving Search Agent

Build an agent that logs failures, analyzes them with a 'Judge LLM', and automatically updates instructions to avoid future mistakes.

Phase 7: Production & Deployment

Goal: Scaling and Securing the system for Banks and Pharma.

Key Topics:

- Local Inference (Ollama/vLLM): Running 32B+ models on private hardware.
- Quantization: GGUF/EXL2 formats for smaller GPU footprints.
- Concurrency & Guardrails: Handling 50+ users without data leaks.
- ROI Calculation: Building the business case for $100k contracts.

Capstone Project: The Private Enterprise 'Brain'

The Graduation Project: Deploy a local, Agentic RAG system on a private server handling 1,000+ docs with 95% RAGAS accuracy.

Phase 8: 1. The Core Language & Logic

Goal: Master the foundational tools for deterministic, structured AI development.

Python (Expert Level)

Master Asyncio for parallel AI calls and Pydantic for structured JSON outputs. This is how you force the AI to follow your rules.

SQL

Many documents are actually database rows. You must build agents that can write and execute their own SQL queries.

This is the most important part. Poor parsing = Project failure.

Docling (by IBM)

Currently the best tool for converting complex PDFs and tables into clean Markdown.

Unstructured.io

The go-to for 'SharePoint hell' (handling .docx, .pptx, and .pdf simultaneously).

Marker

High-speed, high-accuracy PDF-to-Markdown conversion for massive document sets.

LangGraph

The industry standard for Agentic RAG. It allows stateful loops where the AI can 'go back' if a search fails.

LlamaIndex

Specifically Workflows and Query Engines. Superior to LangChain for deep, technical data retrieval.

Qdrant

Highly recommended for enterprise metadata filtering and hybrid search capabilities.

PGVector (PostgreSQL)

Ideal for banks that already use Postgres. Build AI on top of their existing, secure infrastructure.

Ollama & vLLM

Essential for local development (Ollama) and production-grade local hosting (vLLM) for privacy-sensitive clients.

Models to Master

Qwen 2.5 (32B/72B) for technical/math tasks. Llama 3.1 (70B) for high-level reasoning.

RAGAS & DeepEval

Frameworks to grade 'Faithfulness' and 'Answer Relevancy' to prove to clients the AI isn't lying.

Chainlit & Streamlit

Rapidly build professional chat interfaces and data dashboards in pure Python.

Summary Checklist: Your 'Start Today' Stack

1. Framework: Learn LangGraph.
2. Data Extraction: Learn Docling.
3. Local LLM: Install Ollama and run Qwen 2.5.
4. Database: Start a local Qdrant instance via Docker.