# Phase 3 – Retrieval-Augmented Generation (RAG)

## 🔹 RAG Architecture

1. Ingestion Layer  
- Document upload, parsing, and chunking (PDF, DOCX, Markdown, SQL text).  
- Clean text storage + chunking by semantic boundaries.  
  
2. Embedding & Storage Layer  
- Generate vector embeddings (OpenAI or HuggingFace).  
- Store in Postgres + pgvector or Azure Cognitive Search.  
- Maintain metadata in SQL (docId, chunkId, tags, owner, version).  
  
3. Retrieval Layer  
- Semantic + hybrid search over embeddings.  
- Ranking, deduplication, filters.  
  
4. Augmentation + Generation Layer  
- Retrieved chunks + user query → LLM.  
- Compare baseline LLM vs RAG-enhanced responses.  
- Debug mode: show retrieved chunks in Admin Panel.

## 🔄 Phase 3 Sub-Phases

Phase 3.1 – Document Ingestion & Chunking  
- Admin panel document upload.  
- Parsing (PDF/DOCX/MD → text).  
- Store text + chunks in SQL.  
  
Phase 3.2 – Embeddings & Vector Storage  
- Generate embeddings.  
- Store in pgvector or Azure Cognitive Search.  
- Link embeddings with metadata.  
  
Phase 3.3 – Retrieval Service  
- Build IRetrievalService.  
- API endpoint /api/retrieval/search.  
- Top-k semantic search results.  
- Admin Panel: show retrieval debug info.  
  
Phase 3.4 – RAG Pipeline Integration  
- Build IRagService.  
- Input: user query → retrieval → augmented prompt → LLM.  
- Compare baseline vs RAG outputs.  
  
Phase 3.5 – Admin Panel Enhancements  
- Add Documents Page (upload, list, delete, version).  
- Add RAG Comparison Page (baseline vs RAG, chunk debug view).  
  
Phase 3.6 – Deployment & Optimization  
- Deploy vector DB (Azure or pgvector).  
- Optimize retrieval (indexes, caching).  
- Add monitoring (queries/sec, storage growth).

Phase 3 RAG Architecture Diagram:

## ⚡ Expected Outcomes

- RAG-enabled chatbot in .NET + AdminLTE project.  
- Admin panel for document + RAG debugging.  
- Comparison: baseline vs RAG-enhanced responses.  
- Scalable + production-ready retrieval pipeline.

