Day 6 — Enterprise Hybrid RAG + Planning Agent

Date: 2026-02-16

1. What We Built (Simple Explanation)

We built a production-style Hybrid RAG Agent. It plans tasks using structured JSON, retrieves knowledge from a local vector store (RAG), and falls back to Web search when confidence is low. Routing is based on cosine similarity score, not guesswork.

2. Core Architecture

User Goal → Planner (JSON Steps) → HybridToolWrapper → RAG (VectorStore + Embeddings) OR WebSearchTool → Observations → LLM Synthesis → Final Answer.

3. Important Technical Changes Today

- RAGSearchTool now always returns lists of docs and similarity scores.
- VectorStore exposes `documents` and `document_embeddings`.
- Hybrid routing uses similarity threshold (default 0.50).
- top_k=3 retrieval for stable scoring.
- Removed weak length-based routing logic.
- Added confidence logging for transparency.

4. How Routing Works

1. Query → compute embeddings.
2. Compare against vector store using cosine similarity.
3. If max_score >= threshold → use RAG.
4. Else → fallback to Web search.
5. Merge results and synthesize final answer.

5. Why This Is Enterprise-Level

- Deterministic JSON planning.
- Confidence-based retrieval gating.
- Tool abstraction layer.

- Clear separation of planner, retriever, and executor.
- Debug logs for observability.
- Secrets isolated in .env (not pushed to GitHub).

6. How To Verify System

Test 1: 'what are embeddings' → should use RAG (high similarity).
Test 2: 'healthcare updates 2026' → low similarity → fallback to Web.

7. ATS Resume Line

Designed and implemented an enterprise-grade hybrid RAG agent with similarity-gated retrieval, structured JSON planning, persistent vector store, and web fallback integration.

8. How To Start Day 7

Open a new chat and type: lets start with day 7.
All further learning and code will follow enterprise architecture standards.