

14-Day Roadmap: Full-Featured RAG System with LangGraph + CRAG

Day 1: RAG + LangGraph Setup

Understand RAG, install dependencies, set OpenAI key. Resources: FreeCodeCamp RAG video, LangGraph docs.

Day 2: LangGraph Basics + State Machine

Build a dummy graph with 2-3 states and transitions. Understand State, Node, and Flow.

Day 3: Document Loading + Indexing

Ingest and index PDFs using PyMuPDF or PDFPlumber, chunk with RecursiveCharacterTextSplitter, store with FAISS.

Day 4: Retrieval Node

Implement the retrieval logic using langchain retriever. Return top-k relevant documents.

Day 5: Evaluation Node (CRAG logic)

Use LLM to score retrieved docs: Correct, Ambiguous, Incorrect. Route accordingly.

Day 6: Answer Generation + Decompose/Recompose

Generate final answers from refined context. Decompose facts then recompose for answer.

Day 7: Connect All in LangGraph

Connect all states in LangGraph, test with basic queries. Save as ``basic_crag_graph.py``.

Day 8: Multi-Input Upload (PDFs, URLs, Images, Audio)

Enable PDF, URL, image (OCR), audio (Whisper) uploads. Normalize to plain text and embed.

Day 9: Memory Integration

Add session-level memory using LangChain's ConversationBufferMemory to retain chat context and

14-Day Roadmap: Full-Featured RAG System with LangGraph + CRAG

highlights.

Day 10: Summarization + Highlighting

Let users summarize and highlight documents, saving them to memory for future reference or citations.

Day 11: Document Comparison Tool

Compare two or more docs using side-by-side summary and LLM-driven comparison.

Day 12: Web UI with Streamlit or Gradio

Build a Streamlit/Gradio UI with file upload, chat interface, highlighting, and doc comparison.

Day 13: Evaluation + Fine-tuning

Test ambiguous queries, analyze CRAG vs RAG output, integrate evaluators like TruLens.

Day 14: Polish + Deploy

Deploy the app to HuggingFace Spaces, Render, or Railway. Create README, push to GitHub, and polish UX.