Build a Company-Specific RAG Pipeline

Mainsail AI Launch Labs

Reference Material

Before starting, review the RAG Cookbook for examples of different RAG patterns. You are encouraged to use it as inspiration for architecture and tradeoffs, not as a strict template.

Overview

In this assignment, you will design and implement a Retrieval-Augmented Generation pipeline using real documentation from your company.

The goal is to practice turning unstructured company knowledge into a reliable retrieval system and to evaluate how well your retrieval pipeline performs.

This is an applied systems assignment focused on design decisions, retrieval behavior, and evaluation, not UI or polish.

Due Date

Thursday, January 22nd at 11:59 PM CT

Dataset Requirements

- Use **company-specific documents**, such as:
    - Internal wiki or SOPs
    - API or developer documentation
    - Product docs
    - Engineering blogs or knowledge bases

- **Target size:** 50 to 100 documents
- Dataset quality is **not graded**. Focus on structure and retrieval behavior.

If documents are sensitive, you do not need to share the raw data with us.

RAG Pipeline Requirements

You must implement a working RAG pipeline that includes:

1. Document ingestion
2. Chunking strategy
3. Embedding and indexing into a vector database
4. Retrieval
5. Generation using retrieved context

Vector Database

You may use any vector database.

Recommended options:

- MongoDB Atlas Vector Search
- Pinecone

Framework

You may use any framework, including:

- LangChain
- Vercel AI SDK
- OpenAI Agents SDK
- Anthropic Agents SDK
- A custom or lightweight implementation

No client or UI is required. The system can run locally, via scripts, or as an API.

RAG Pattern Requirement

Implement **at least one** RAG pattern:

- Naive RAG
- Metadata-filtered RAG
- Hybrid search
- Graph RAG
- Agentic RAG

You should clearly explain which pattern you chose and why it fits your documents and use case.

Eval Requirement

You must implement **at least one retrieval-based eval**.

Allowed evals include:

- Precision
- Recall
- Groundedness

Advanced ranking metrics like MRR or nDCG are **not required**.

The eval must explicitly measure retrieval behavior, not just the final LLM response.

1. Code

Submit code that shows how your RAG pipeline is constructed, including:

- Ingestion and chunking logic
- Retrieval logic
- Eval logic

The code does **not** need to be fully runnable end-to-end, since you may not want to expose internal documents or credentials. Clarity and structure matter more than execution.

2. Video Walkthrough

Submit a **3 to 5 minute video** explaining:

- How your pipeline is structured
- Your chunking and retrieval decisions
- The RAG pattern you implemented
- The eval you chose and why

A live demo is optional.

Grading Criteria

- Clear RAG pipeline design
- Correct use of a RAG pattern
- Proper implementation of a retrieval-based eval
- Thoughtful reasoning about tradeoffs
- Code clarity and organization

Dataset quality, UI, and deployment are **not** part of grading.

Ready to face the Gauntlet?

Gauntlet AI - Where only the strongest AI-first engineers emerge.