Task 2: Retrieval + Guardrails – Lean FastAPI

What to Ship First:

For this task, the first thing I'd focus on is building the **core FastAPI application** with the `/answer` endpoint. This endpoint should handle incoming questions, retrieve the top documents from our knowledge base using the **cosine similarity** method (this will be our default search method), and return the most relevant information to the user. I'd also implement a simple **guardrail** to block sensitive queries, like anything related to **salary** or **termination**.

The **naive answer generation** logic would be included as well, so users get quick responses based on the top snippets retrieved from the search. The aim here is to get a fully functioning **retrieval system** that can handle basic queries and provide answers quickly.

Additionally, I'd ship the basic **metrics tracking**, so we can keep an eye on how many requests are being processed and how often sensitive queries are blocked. It's also good to start tracking **response time**, even if it's a placeholder for now.

What to Ship Later:

Once the basic functionality is in place, there are several things we can work on to improve the system. First, we should definitely look into improving the **search methods**. Cosine similarity works great, but if we scale up the corpus, we may want to explore more **advanced retrieval techniques** like using **FAISS** or another vector search engine. These would give us better speed and accuracy as we deal with more data.

We could also improve the **guardrails** over time. Right now, we're using a simple denylist for sensitive topics, but eventually, we might want to make it more intelligent by incorporating things like **content moderation tools** or AI-based filtering to better identify problematic queries.

In the long run, having a solid **monitoring system** will be key. As we scale up, we should implement more detailed **latency tracking**, keep an eye on the **hit rate** (how often the top-k snippets are relevant), and track how the model's answers are drifting over time.

Finally, with more time, it would be great to handle larger corpora better. As the knowledge base grows, we'll want to ensure that the system can search faster without compromising quality, so **dynamic indexing** will be something to look into as we move forward.