Assignment for Lead Gen AI Engineer

The Challenge

Build a conversational AI assistant using one of the following options:

Option 1: Chat With Your Docs
Build a system that answers questions about content from a document collection (PDFs, text files, or any format you choose). This is the same classic RAG use-case you might be familiar with.

Option 2: Code Documentation Assistant
Build a system that ingests a codebase (GitHub repo or local files) and answers questions about the code. For example: how it works, where functionality is implemented, API endpoints, dependencies, etc.

Option 3: Meeting Intelligence System
Build a system that analyzes meeting transcripts and answers questions about discussions, decisions, and action items. Transcripts should be text files with speaker labels and timestamps. Optional bonus: Add voice-to-transcript capability.

Option 4: Career Intelligence Assistant
Build a system that analyzes resumes against job descriptions. Upload a resume and multiple job postings, then answer questions about fit, skill gaps, experience alignment, and interview preparation.
Example queries: "What skills am I missing for this role?", "How does my experience align with Job #2?"

Choose the option that excites you most!

We care as much about *how* you build it as *what* you build. This is your chance to show us your engineering philosophy in action.

Creative Freedom

You decide the tech stack:

- Use case selection & sataset/codebase/meeting content

- Programming language (Python, JavaScript, Java, etc)
- LLM provider (OpenAI, Anthropic, Ollama, etc)
- Vector database and Orchestration Framework
- Architecture (monolith, microservices, serverless, etc)

What to Submit

1. **GitHub repo** with your code
2. **README.md** / other docs in the repo with:
    a. Quick setup instructions
    b. Architecture overview (a simple diagram is great but not required)
    c. What would be required to productionize your solution, make it scalable and deploy it on a hyper-scaler such as AWS / GCP / Azure?
    d. RAG/LLM approach & decisions: Choices considered and final choice for LLM / embedding model / vector database / orchestration framework, prompt & context management, guardrails, quality, observability
    e. Key technical decisions you made and why
    f. Engineering standards you've followed (and maybe some that you skipped)
    g. How you used AI tools in your development process
    h. What you'd do differently with more time
    i. Note: We need your thoughts, not an LLM's direct output

What We're Looking For

- **Core Functionality:** A working solution that can answer questions based on uploaded/provided documents using RAG or similar retrieval mechanisms with a simple interface
- **Your approach and thought process towards:** Chunking, Embedding Model & LLM Selection, Retrieval Approach, Prompt Engineering, Context Management, Guardrails, Quality Controls, Observability
- **Engineering excellence**: Clean, readable, well-structured code that others can understand. Ideally a simple containerised, well tested and observable solution.
- **Development approach:** You're free to AI Coding tools of your choice. We're interested in how you use these AI coding tools and ensure that the code is according to your preference? How do you make it repeatable and maintainable? What are your do's and don'ts with AI coding assistants?

Tips & Notes

- **We're not looking for perfection** - we're looking for how you approach problems, make trade-offs, and apply best practices within realistic time constraints.
- **Start simple, then enhance** - A basic working version with great engineering > a complex broken version
- **We won't judge you on:**
    - Perfect UI/UX (we care more about the approach)
    - Handling every edge case (acknowledge them in your README)
    - Using the "best" tech stack (we care about your reasoning)
- **We want your thoughts and ideas (not an LLM's outputs)** - While you're free to use AI coding assistants, we want your readme file to reflect your thought process and we expect you to judge where it's okay to use LLMs outputs and where you need to articulate things yourself
- **Not enough time to implement everything?** That's okay! Document what you'd add next in your README.

Remember, we value a solid & well-engineered basic solution A LOT MORE than an over-engineered complex one.