Applied AI & Machine Learning Assessment

Build Your Own RAG System

Objective

Build a **Retrieval-Augmented Generation (RAG) architecture**, integrating a **vector database** and an **embedding model**, to power the backend of a **Q&A application**. The system should retrieve relevant knowledge from the provided files and generate accurate responses.

Guidelines

✅ **Choose Any Vector Database** – You are free to use any vector DB (e.g., FAISS, Pinecone, Chroma, Weaviate, etc.).
✅ **Clean & Documented Code** – We will run your code in our environment, so ensure it's well-structured, documented, and easy to execute.
✅ **Runtime Considerations** – Ensure your code runs within a reasonable time on standard hardware.

Important Notes

🚨 **API Access** – We have provided an endpoint to GPT, which will be removed after the assessment. We are monitoring its usage, so **do not share or abuse the access**.
📊 **Presentation Requirement** – Along with the code, please prepare a **2-3 slide deck** explaining your solution, architecture, and approach.

After submitting your assignment, we'll have a follow-up interview where we'll **deep dive into your solution**

If you have any questions or need clarification during the assessment, feel free to reach out.

📧 **Email:** oarman@kpmg.com
📧 **Email:** yisraeli@kpmg.com

We're here to help—don't hesitate to ask!
good luck! 🚀

Checklist for a Successful Submission

Assessment Resources	• **Data Files:** included in the provided ZIP file • **API Credentials:** azure_endpoint ='https://interviews3.openai.azure.com/', api_key ='5UnXrfATc5KyXEVyxpjeJF9MInBazuoBMBkyHKEB1nARFKxuJGLtJQQJ99BCAC4f1cMXJ3w3AAABACOGNEeL', api_version="2024-08-01-preview", model = 'gpt-35-turbo'
Expected Deliverables	• A **Python-based RAG implementation** • A **2-3 slide deck** explaining the approach • A **README file** with setup instructions and assumptions
Submission Format	• Please submit your code via a GitHub repo or a ZIP file with all necessary files • We appreciate clean, modular code with well-structured comments and docstrings
Pro Tips	• **Explain Your Thought Process** – In your slide deck, highlight your design choices, any trade-offs made (speed, memory usage, and accuracy), and alternative approaches you considered. • **Test with Edge Cases** – Think about how your system handles ambiguous queries, large documents, or missing data. A few well-placed tests can make a big difference.