Approach

The solution leverages a Retrieval-Augmented Generation (RAG) pipeline to recommend SHL assessments based on job descriptions or URLs. The system combines semantic search using vector embeddings with large language model (LLM) reasoning to generate context-aware recommendations.

Key Steps

Data Collection & Preparation

- Web Scraping: Custom Python scripts (shl_scraper.py) extract assessment details (title, URL, remote testing support, adaptive/IRT status, test type, and details) from SHL's product catalog.
- Embedding Generation: Mistral's mistral-embed model converts assessment descriptions into 768-dimensional vectors.
- Vector Database: Pinecone stores embeddings and metadata for fast similarity searches.

IMPORTANT NOTE: Since there are 600 assignments in SHL's catalog, I only scraped 120 assignments for this demo as scraping and storing 600 assignments was not convenient. A csv file containing the details of those 120 assessments is on the github repo.

RAG Pipeline

- Input Handling: Users provide either a job description text or a URL (scraped using BeautifulSoup).
- Semantic Search: Mistral embeddings enable similarity matching between the job description and stored assessments.
- LLM Refinement: The top 10 results are formatted into a markdown table using Groq's deepseek-r1-distill-llama-70b model to ensure readability and relevance.

Deployment

- Streamlit Web App: Hosted on Streamlit Share for user interaction.
- FastAPI Endpoint: Provides a JSON API for programmatic access.

Tools & Libraries

Scraping: BeautifulSoup, requests
Embeddings: MistralAI embeddings
Framework: Langchain, Langsmith for tracing
Tracing: Langsmith
Vector DB: pinecone
Model: deepseek-r1-distill-llama-70b from groq
Frontend: Streamlit
API: FastAPI, Uvicorn