Assessment Recommendation System

Objective - Hiring managers often struggle to find the right assessments for the roles that they are hiring for. The current system relies on keyword searches and filters, making the process time-consuming and inefficient. Your task is to build an intelligent recommendation system that simplifies this process. Given a natural language query or a job description text or URL, your application should return a list of relevant SHL assessments. You can look at the data sources that you are going to work with here, https://www.shl.com/solutions/products/product-catalog/

Tech Stack & Tools

- **FastAPI**: Lightweight Python web framework for serving endpoints (/search, /status, etc.)
- **LlamaIndex**: Framework for building LLM-based query pipelines over structured and unstructured data
- **FAISS**: Efficient similarity search engine for vector indexing
- **HuggingFace Transformers**: For generating semantic embeddings using all-mpnet-base-v2
- **Groq API**: Hosted LLaMA 4 model used for intelligent query understanding
- **SQLite**: Lightweight database used to store SHL assessment metadata
- **dotenv**: Manages API keys and secrets securely using environment variables

Architecture & Workflow

1. **Data Preparation**
   The SHL product catalog CSV is loaded into a **SQLite** database (assessments.db) via db.py.
2. **Embedding & Indexing**
   Using **HuggingFace's all-mpnet-base-v2**, each assessment's textual information is embedded and stored in a **FAISS** index (vector_index_faiss). This is handled in index_builder.py.
3. **Semantic Query Engine**
   Queries are processed via **LlamaIndex**'s VectorStoreIndex. The engine uses the **Groq-hosted LLaMA 4** model to improve semantic understanding of user queries. Top results are filtered based on vector similarity and enriched with corresponding data from the SQLite database.
4. **API Endpoints (FastAPI)**
   - POST /search: Accepts a natural language query and returns up to 10 relevant assessments.
   - GET /status: Performs health checks on vector index, database, and key environment variables. Returns {"status": "healthy"} if all components are operational.
   - GET /: Serves a basic HTML template as a landing page.

Security & Deployment Considerations

- API keys (like GROQ_API_KEY) are **never hardcoded**. They're loaded securely using .env and python-dotenv.
- .env is excluded from version control to prevent credential leaks.

Outcome

This system enables fast, accurate, and scalable semantic search over structured assessment data, using cutting-edge NLP models and vector search libraries. It is production-ready with robust health checks and a secure configuration setup.