GenAI Assessment Recommendation

Building a web-based RAG tool using SHL product catalogue

Chirag Ajay Jain | 10th November, 2025 | GitHub Repo: github.com/ChiragAJain/SHL-Hiring-Assessment

Live App: shl-assessment-api-production.up.railway.app

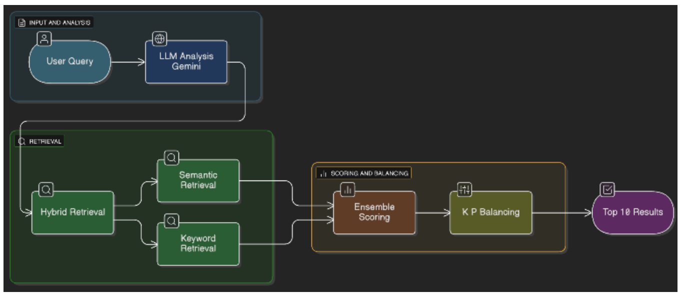
I. Introduction

Problem Statement: Hiring managers and recruiters struggle to find the right assessment tools from SHL's extensive catalogue. The current system relies on keyword searches, making the process time-consuming and inefficient for users who may not know the exact assessment name for a given job's requirements.

Solution: This project delivers an AI-powered recommendation system that accepts a natural language query or full job description. It analyses the text, understands its requirements, and returns a balanced list of the top 10 most relevant "individual test solutions" from the SHL catalogue, directly addressing the core problem of inefficient search.

II. Technical Architecture

The system is a "Retrieve & Re-rank" pipeline built on a FastAPI backend. It uses a Hybrid Search model to gather a wide pool of candidates and an Ensemble Scoring algorithm to rank them for relevance.



Architecture Diagram

Technology Stack:

- **Backend:** Python 3.10, FastAPI, Uvicorn
- **LLM Query Analysis:** Google Gemini
- **Vector Database:** ChromaDB (v0.5.23)
- **Embedding Model:** E5-large-v2 (1024 dimensions)
- **Deployment:** Railway (8GB RAM)

III. Data Pipeline

1. **Scraping:** A custom Selenium-based scraper was built to navigate the SHL product catalogue's dynamic JavaScript rendering and pagination.
2. **Extraction:** The scraper collected 377 unique "individual test solutions," extracting their Name, Description, URL, Test Types (K, P, A, etc.), Skills, and Job Levels. "Pre-packaged Job Solutions" were ignored as per the assignment.
3. **Storage:** The cleaned and structured data was saved as shl_assessments.json, which serves as the "source of truth" for the keyword search and API responses.
4. **Indexing:** All 377 assessments were embedded using the E5-large-v2 model and indexed into a persistent ChromaDB vector database to power semantic search.

IV. Implementation: Hybrid Search & Ensemble Scoring

To achieve high relevance, a hybrid approach was necessary. The system retrieves two distinct pools of candidates which are then merged and re-ranked.

1. Semantic Retrieval (Pool A): The LLM-expanded query is used to find the top 50 *conceptually* similar assessments from the ChromaDB vector store. This pool is good at understanding intent (e.g., "assessment for a leader").
2. Keyword Retrieval (Pool B): The raw skills extracted by the LLM (e.g., "Java," "Python," "SQL") are used to perform a literal keyword search against the shl_assessments.json file. This pool is essential for finding *exact* skill matches (e.g., "Python (New)").

The two pools are merged, and every unique assessment is scored using a weighted ensemble algorithm:
Final Score = (35% * Semantic) + (45% * Keyword) + (20% * Metadata)

- Semantic Score (35%): The raw cosine similarity score from the E5-large-v2 model.
- Keyword Score (45%): A heavily weighted score based on literal matches to the LLM-extracted skills.
  - 5.0x weight: Exact match for a programming language (e.g., "Java").
  - 4.0x weight: Match for a job role (e.g., "Analyst").
  - 3.0x weight: Match for a soft skill (e.g., "collaboration").
- Metadata Score (20%): A boost for aligning with the query's job_level or duration constraints.

Finally, a Smart K/P Balancing algorithm ensures the top 10 results contain a healthy mix of "Knowledge & Skills" (K) and "Personality & Behavior" (P) tests, as specified in the assignment.

V. Performance & Evaluation

The system was evaluated against a manually-labelled ground truth dataset of 90 real-world job queries. The primary metric was Mean Recall@10, as defined in the assignment.

Metric	Score
Mean Recall@10	**23.33%**
Mean Recall@50	43.33%
Mean Recall@100	46.44%

Response Time:

- Cold Start: ~30 seconds (for E5-large-v2 model to load into memory).
- Subsequent Requests: < 1 second (average).

VI. Development Journey & Optimisation

The final 23.33% recall score was achieved after several key experiments and optimisations.

Initial Approach: Pure Semantic Search

My first iteration used only the E5-large-v2 semantic search. This performed poorly (<15% Recall@10).

- Finding: I discovered the ground truth labels were literal, not semantic. For a query asking for "Python" and "SQL," the correct answers were the literal "Python (New)" and "SQL Server (New)" tests, which a pure semantic search failed to find, retrieving "Data Warehousing Concepts" instead.

Experiment 1: Model Context Protocol (MCP) Server

I hypothesized that giving the LLM direct control over search tools would be more "intelligent."

- Implementation: I built a full MCP server wrapper around the search tools.
- Result (Failure): This approach was abandoned. It added 2-3 seconds of latency, produced inconsistent search strategies, and made debugging impossible.
- Lesson: LLMs are excellent for *query analysis* (understanding intent), but a deterministic, weighted algorithm is far superior for the *ranking* process.

Experiment 2: Hybrid Search & Ensemble Scoring (The Solution)

Based on the failure of the pure semantic model, I built the hybrid system.

- Implementation: I combined a literal keyword search (to find the *exact* matches from the ground truth) with the semantic search (to understand context).
- Optimisation: I iteratively tuned the ensemble weights, testing various combinations. A "keyword-heavy" weighting of 35% Semantic / 45% Keyword / 20% Metadata yielded the best results, as it correctly prioritized the critical, literal skill matches while still accounting for the query's overall intent.

Challenge: Cold Starts & Memory

The 1.3GB E5-large-v2 model caused 30+ second cold starts and exceeded the memory limits of free hosting tiers.

- Solution: I migrated the deployment to Railway on an 8GB RAM plan, which could hold the model in memory. This resulted in sub-second response times after the initial (unavoidable) cold start.