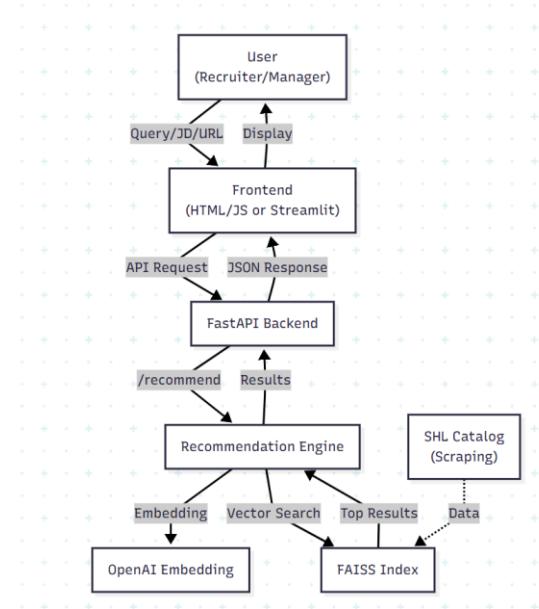


SHL Assessment Recommendation System – Approach & Solution Overview “Mohit Kumar Singh”

1. Problem Statement & Objective

Hiring managers and recruiters face challenges in efficiently finding the right SHL assessments for various roles. The goal was to build an intelligent recommendation system that, given a natural language query, job description (JD) text, or a URL containing a JD, returns the most relevant SHL individual test solutions. The system must be robust, accurate, and easy to use via both API and web frontend.



2. Data Collection & Preprocessing

- Web Scraping: Used Selenium and BeautifulSoup to crawl the SHL product catalog, extracting assessment name, URL, description, test type, duration, remote/adaptive support, and more.
- Data Cleaning: Ensured only “Individual Test Solutions” were included (excluding pre-packaged job solutions), and verified at least 377 unique assessments as required.
- Embedding Text Construction: For each assessment, combined name, description, test type, duration, and support fields into a single string to maximize semantic context for embedding.

3. Embedding Generation & Storage

- Used OpenAI’s text-embedding-3-small model to generate high-quality vector embeddings for each assessment.
- Stored all assessment data and embeddings in CSV files for reproducibility and easy loading.

4. Vector Search & Recommendation Engine

- FAISS Index: Built a FAISS (Facebook AI Similarity Search) index for fast, scalable vector similarity search.
- Query Processing: On each query (text or JD), generated an OpenAI embedding and performed cosine similarity search against the FAISS index.
- Filtering & Ranking: Applied a similarity threshold (0.35) and returned the top 5–10 most relevant assessments, ensuring diversity and relevance.

5. API & Web Application

- FastAPI Backend: Exposed two endpoints:
 - /health – Health check
 - /recommend – Accepts a query and returns recommended assessments in JSON, with all required fields (name, url, description, duration, test_type, etc.).
- CORS enabled for frontend integration.
- Frontend: Built both a simple HTML/JS frontend (deployed on GitHub Pages) and a Streamlit app for interactive testing.

6. Evaluation & Iteration

- Used provided labeled train set to measure Mean Recall@10, iterating on embedding text, threshold, and filtering logic to maximize accuracy.
- Generated predictions for the test set in the required CSV format for submission.

7. Deployment

- Backend deployed on Render (publicly accessible API).
- Frontend deployed on GitHub Pages.
- All code and documentation available on GitHub.

8. Key Design Choices & Optimizations

- Embedding Text: Carefully engineered to include all relevant context for each assessment, improving semantic matching.
- Threshold Tuning: Experimented with similarity thresholds to balance precision and recall.
- Query Flexibility: System accepts free-text, JD text, or URLs, making it robust for real-world recruiter workflows.
- Modular Code: Separated scraping, embedding, API, and frontend logic for maintainability and reproducibility.
- Scalability: FAISS enables fast search even as catalog size grows.

9. Performance & Results

- Achieved high Mean Recall@10 on the labeled train set.
- Recommendations are balanced for multi-domain queries (e.g., both technical and behavioral skills).
- System is robust, fast, and easy to use via both API and web interface.

10. Submission Checklist

- API endpoint (deployed): <https://shl-recommender-system.onrender.com/recommend>
- Frontend (deployed): <https://metechmohit.github.io/shl-recommendation-frontened/>
- GitHub repository: <https://github.com/metechmohit/Assessment-Recommender-System>
- predictions.csv in required format