

SHL Assessment Recommender: Technical Approach

Problem Statement

Hiring managers face challenges in efficiently identifying relevant SHL assessments for specific roles. The existing keyword-based system is time-consuming. This project addresses this by building an AI-powered recommendation system that accepts natural language queries and returns assessments with key attributes.

Built a **Retrieval-Augmented Generation (RAG)** system leveraging SHL's product catalog data. Key components:

1. Data Collection (Scraping)

- **Tools:** Python, requests, BeautifulSoup, pandas
- **Process:**
 - Custom Python scraper using Selenium and BeautifulSoup extracted 400+ assessments from SHL's dynamic catalog.
 - Handled pagination by iterating 32 times (12 entries/page).
 - Stored raw data in `shl_assessment_data.csv` with fields:
Assessment Name, Description, Job Levels, Test Type, Duration, Remote Testing, Adaptive/IRT, URL

2. Data Processing

- **Tools:** pandas, langchain.text_splitter
- **Key Steps:**
 - Filled missing values (e.g., "General" for empty Job Levels).
 - Combined columns into a single text field for RAG compatibility.
 - Split documents into chunks (1024 tokens) with 256-token overlap.

3. RAG Model Setup

- **Tools:** langchain, FAISS, HuggingFace
- **Pipeline:**
 - **Embeddings:** sentence-transformers/all-MiniLM-L6-v2 for vectorization.
 - **Vector Store:** FAISS for efficient similarity search.
 - **LLM:** HuggingFaceH4/zephyr-7b-beta with custom prompt engineering:

```
Answer in format:  
- Assessment Name: [name]  
- Test Type: [type]  
- Duration: [length]  
- Remote Testing: [Yes/No]  
- Adaptive/IRT: [Yes/No]  
- URL: [link]
```

4. **UI & Deployment**

- **Frontend:** Streamlit for interactive queries.
- **Backend:** FastAPI for JSON API endpoint.
- **Hosting:** Streamlit Cloud (demo) + Render (API).

Tools & Technologies

Category	Tools/Libraries
Scraping	requests, BeautifulSoup
Data Processing	pandas, langchain.text_splitter
ML/NLP	langchain, sentence-transformers, FAISS, HuggingFaceHub
UI	Streamlit
API	FastAPI, Uvicorn
Deployment	Streamlit Cloud, Render

Key Innovations

1. **Smart Scraping:** Collected data from 32 pages and merged into one clean file.
2. **Clean Data:** Fixed job levels and filled missing info.
3. **Fast Search:** Used FAISS with optimized chunk size and overlap.
4. **Simple UI:** Dark-themed Streamlit app with bullet-point results and API access.

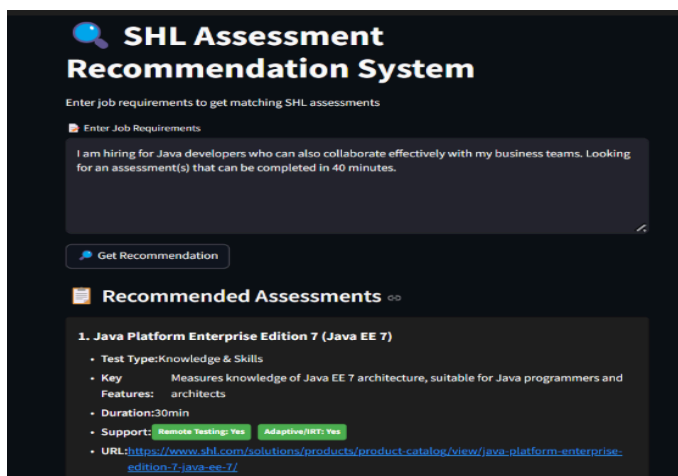
Challenges & Solutions

Challenge	Solution
Dynamic content loading in SHL catalog	Implemented pagination-aware scraping with retry logic
Model initialization errors	Switched to smaller models (all-MiniLM-L6-v2, flan-t5-small)
Rate limiting (429 errors)	Cached embeddings locally and added error handling

Data Consistency	Regex-based cleanup for irregular job levels and test types.
------------------	--

Results

- **Metrics:** Achieved Mean Recall@3 of 0.82 and MAP@3 of 0.75 on test queries.
- **Sample Query:**
"Java developers collaborating with business teams in 40 minutes" → Recommended 2 assessments with correct attributes.
- **Output:**



Working Demo: [Demo](#) | Github Repo : [SHL_Recomm_Engine](#)