**SHL Assessment Recommender - Technical Overview**

**Objective**

To build an end-to-end AI-powered recommendation system for SHL assessments by scraping individual assessment pages, structuring relevant metadata, and applying semantic similarity for personalized matching.

**Project Structure**

* **Backend (FastAPI)**

Located in app/, this includes the API (main.py), data models (models.py), scraping logic (scraper.py), and recommendation engine (recommender.py).

* **Frontend (Streamlit)**

A user interface in frontend/app.py allows users to interact with the recommender system.

* **Data Layer**
  + raw/: Stores unprocessed scraped content (shl\_catalog\_raw.json, shl\_page.html)
  + processed/: Stores structured data (shl\_assessments\_detail.json) with optional embeddings
  + debug/: Contains intermediate files and rendered HTML snapshots for troubleshooting

**Tools and Libraries Used**

* FastAPI - For building a lightweight, asynchronous API
* Streamlit - For the frontend interface
* Playwright - For scraping JavaScript-rendered SHL pages
* BeautifulSoup (bs4) - For parsing rendered HTML and extracting data fields
* Pydantic - For schema validation and structured API models
* Sentence Transformers / OpenAI Embeddings - For vector-based semantic similarity
* FAISS / Chroma (optional) - For scalable similarity search
* NumPy & Scikit-learn - For cosine similarity and evaluation metrics

**Solution Workflow**

1. Scraping and Data Extraction (scraper.py)
   * Uses Playwright to visit each SHL assessment page individually.
   * Extracts fields such as:
     + name, description, duration, job\_levels, languages
     + Boolean flags: remote\_testing\_support, adaptive\_irt\_support
     + Downloadable resources (PDF links)
2. Data Structuring and Cleaning (utils.py)
   * Normalizes fields (e.g., formats durations, splits lists, removes noise)
   * Stores structured results in shl\_catalog\_raw.json and shl\_assessments\_detail.json
3. Embedding & Recommendation Logic (recommender.py)
   * Generates semantic embeddings for assessment descriptions or metadata
   * Computes cosine similarity between user input and stored embeddings
   * Returns top-N most relevant assessments based on similarity scores
4. Evaluation (evaluation/)
   * Includes benchmarking scripts and custom evaluation metrics to assess recommendation

Performance.

1. Frontend (frontend/app.py)
   * Streamlit interface allows users to input skills or job roles
   * Displays matched assessments with key metadata and relevance score

Output Format (example)

{

"name": "Cognitive Ability Test", "duration": "30 minutes", "job\_levels": ["Entry Level"], "languages": ["English", "French"], "remote\_testing\_support": true, "adaptive\_irt\_support": false, "description": "...",

"pdf\_link": ["https://www.shl.com/resource.pdf"](http://www.shl.com/resource.pdf)

}