SHL Assessment Recommendation System – Approach Document

1. Problem Understanding

Hiring managers often struggle to identify the most relevant assessments for a given role using keyword-based search systems. The goal of this project is to design an intelligent recommendation system that accepts a natural language query or job description and returns 5–10 relevant SHL Individual Test Solutions. The system must be scalable, accurate, and capable of balancing technical and behavioral assessments as required by SHL.

2. Data Collection and Processing

The first step involved scraping the SHL Product Catalog, focusing exclusively on Individual Test Solutions while excluding pre-packaged job solutions. For each assessment, attributes such as name, URL, description, test type, duration, remote support, and adaptive support were extracted. The cleaned data was stored in a structured CSV format and validated to ensure coverage of more than 377 assessments as required.

3. Recommendation Architecture

To overcome the limitations of keyword search, a semantic retrieval-based approach was implemented. Each assessment description was converted into dense vector embeddings using a SentenceTransformer model. These embeddings were indexed using FAISS to enable efficient similarity search. User queries are also embedded and matched against the index to retrieve the most relevant assessments.

4. Balancing Logic

A post-retrieval balancing layer was added to ensure the final recommendations include a mix of Knowledge & Skills tests and Personality & Behavioral assessments. This directly aligns with SHL’s requirement to provide balanced recommendations when a role spans multiple competency domains.

5. API and Application Design

The system is exposed via a FastAPI backend with two endpoints: a health check endpoint and a recommendation endpoint. The API strictly follows the response format specified in the SHL assessment document. A Streamlit-based frontend allows users to input queries and view recommendations in a user-friendly table format.

6. Evaluation and Optimization

To evaluate performance, the provided labeled training dataset was used to compute Recall@10. Initial experiments were conducted using basic similarity retrieval, followed by improvements through embedding tuning and balancing logic. These steps resulted in improved recall and more diverse recommendations. Final predictions were generated for the unlabeled test set and submitted in the prescribed CSV format.

7. Conclusion

This project demonstrates a complete end-to-end recommendation system using modern retrieval-augmented techniques. The final solution satisfies all technical and functional requirements outlined by SHL, including data scraping, semantic search, evaluation, API exposure, and usability.