**SHL Assessment Recommendation Engine – Approach Document**

**Overview**

This project was created as a part of the SHL AI Intern Evaluation Assignment. The main goal was to create an engine that accepts a natural language inquiry (e.g., a job role or profile description) and provides suggested SHL tests matching appropriate competencies and roles. This project enabled me to integrate NLP principles, API design, and real-time interaction into an efficient recommendation system.

**My Approach**

1. Understanding the Problem: The difficulty was to interpret free-text input and meaningfully map it onto SHL tests through roles and competencies. This meant parsing the natural language input of the candidate and logically correlating it with structured assessment data.
2. Data Extraction & Structuring: Employed BeautifulSoup to scrape the initial role, competency, and assessment data from HTML files. Parsed and structured the data into Python objects, which made them queryable.
3. Recommendation Engine Logic: Pre-processeds the natural language query with simple cleaning and tokenization. Merged role titles and competency names into a single corpus for matching. Used TF-IDF vectorization (from scikit-learn) to transform text data into feature vectors. Used cosine similarity to calculate the similarity between the query vector and all role vectors. Chose the top N similar roles, summed their corresponding competencies, and suggested assessments that match these competencies. Ranked tests by the number of overlapping competencies and returned the best results.
4. API Development & UI: Constructed a REST API using FastAPI to process incoming queries and respond. Utilized Streamlit to implement a light interactive frontend, where users can enter their query and get the recommendations immediately. Made sure that the system is modular, easy to maintain, and easy to extend for future incorporation or deployment of ML models.
5. Tools & Libraries Used

* FastAPI – to implement strong backend API endpoints
* Uvicorn – to host the FastAPI server
* Streamlit – to implement a fast and simple web interface
* BeautifulSoup – for HTML scraping
* Pandas, NumPy – for data wrangling and numerical operations
* Scikit-learn – for vectorization and similarity scoring
* Pydantic – for data validation and model structuring
* Requests, Python-dotenv – for environment handling and HTTP interactions

**Key Learnings**

Learned to connect unstructured user input with structured data using light-weight NLP technique

Got hands-on experience in deploying a full-stack recommendation system. Understood how to make scalable APIs and clean frontends for real-time applications. Enhanced capacity to organize real-world datasets into reusable logic pipelines. Got to learn a lot of new libraries and tools via this project

**Closing Thoughts**

This project forced me to go end-to-end—from user query interpretation to providing meaningful output. The process of scraping raw material, creating a logic-based recommendation engine, and encapsulating it in a functioning web app was extremely rewarding. I hope to further improve this with more sophisticated NLP and more advanced ML methods.