1. Problem Overview

Hiring managers face a significant challenge in efficiently selecting the most appropriate SHL assessments for their job roles. The existing process relies on manual browsing and keyword searches through a large catalog, making it difficult to:

- Find assessments that match complex, multi-faceted job requirements.
- Achieve a suitable balance between technical skill evaluations and behavioral competency tests.
- Quickly identify relevant options from hundreds of choices without a deep contextual understanding of each assessment.

This project addresses these inefficiencies by creating an intelligent recommendation system that accepts natural language queries and provides a balanced, relevant, and well-explained list of 5-10 SHL assessments, saving time and improving hiring decisions.

2. Data Crawling & Preprocessing

The foundation of the system is a comprehensive and clean dataset of SHL's individual assessments.

- **Data Source**: The process begins with a list of URLs from the Gen_AI_Dataset.xlsx file, pointing to specific pages within the SHL Product Catalog.
- **Crawling**: A Python script (scraper.py) using BeautifulSoup and requests visits each URL to extract key information.
- **Filtering**: A critical step in the crawling process is to exclusively target **"individual test solutions"** while explicitly filtering out and ignoring "Pre-packaged Job Solutions" to meet the project requirements.
- **Extracted Fields**: For each valid assessment, the scraper extracts:
    - Assessment Name & URL
    - Detailed Description
    - Assessment Type (e.g., Knowledge & Skills, Personality & Behavior)
    - Applicable Job Levels
    - Supported Languages
    - Assessment Length (completion time)
- **Preprocessing for Retrieval**: Before generating embeddings, all the extracted textual data for a single assessment is concatenated into a single rich text document. This format ensures that the resulting vector representation captures the full semantic context of the assessment, including its purpose, target audience, and constraints. The final structured data is saved to data/shl_assessments.csv.

3. Model / Embedding / Retrieval Method

The core of the solution is a **Retrieval-Augmented Generation (RAG)** architecture that combines fast semantic search with the nuanced understanding of a Large Language Model (LLM).

- **Embedding Model**: The **all-MiniLM-L6-v2** model from the Sentence Transformers library is used to convert both the preprocessed assessment documents and the user's incoming query into 384-dimension vector embeddings. This model was selected for its excellent balance of high performance and computational efficiency.
- **Retrieval Method**:
    1. **Vector Database**: The generated embeddings for all assessments are stored in a **FAISS (Facebook AI Similarity Search)** index. FAISS was chosen for its high speed and memory efficiency, making it ideal for real-time similarity searches.

2. **Semantic Search**: When a user submits a query, it is first converted into an embedding. FAISS then performs a similarity search on this query vector to retrieve the **top 20** most semantically similar assessment candidates from the index.

- **Generation & Re-ranking Model**: The 20 retrieved candidates, along with the original user query, are passed to **Groq's Llama 3.1 8B Instant** model. The LLM acts as an intelligent re-ranker, performing several key tasks:
    1. **Analyzes** the user's query to understand specific requirements like job role, seniority, technical skills, behavioral traits, and time constraints.
    2. **Evaluates** each of the 20 candidates against these specific requirements.
    3. **Selects and re-ranks** the candidates to produce a final, balanced list of the 5-10 most relevant assessments.
    4. **Generates** a concise description and a "Why it's a great fit" explanation for each recommendation, directly linking it to the user's query.
    5. **Formats** the final output into a structured JSON object for reliable use by the backend API.

4. Optimization Steps & Evaluation (Mean Recall@10)

The system's performance was improved through several key iterations, with progress measured using the **Mean Recall@10** metric. This metric is ideal as it evaluates the fraction of relevant assessments found within the top 10 recommendations, reflecting real-world user experience.

- **Baseline Approach**: The initial version used a simple semantic search, returning the top 10 results directly from FAISS.
    - **Limitations**: This method struggled with complex queries, failed to balance technical and behavioral assessments, and provided no explanations.
    - **Estimated Performance**: Mean Recall@10 was estimated to be around **0.45 - 0.55**.
- **Iteration 1: Introducing the RAG Pipeline**: The first major optimization was implementing the RAG pipeline. Retrieving 20 candidates and using the Llama 3.1 8B model for re-ranking significantly improved contextual understanding and the ability to handle more nuanced queries.
    - **Improvement**: The system could now better match assessments to complex job descriptions and generate explanations.
    - **Estimated Performance**: Mean Recall@10 improved to approximately **0.65 - 0.75**.
- **Iteration 2: Advanced Prompt Engineering (Critical Optimization)**: The most significant performance gain came from refining the prompt sent to the LLM. The final prompt gives explicit, rule-based instructions to:
    - Deconstruct the query into technical skills, behavioral traits, and other constraints.
    - **Explicitly balance the results**: The prompt contains a strict instruction: *"If the query mentions both technical abilities and behavioral traits... you MUST ensure your final recommendations include a mix of assessments covering both 'Knowledge & Skills' and 'Personality & Behavior'."*
    - Adhere to a strict JSON output format for reliability.

This advanced prompt engineering directly addressed the core "recommendation balance" requirement and dramatically improved the quality and relevance of the final output.

5. Final Performance Summary

The final version of the system, powered by the RAG architecture and advanced prompt engineering, is highly effective. It successfully provides relevant, balanced, and well-justified recommendations in real-time.

- **High Accuracy**: The system consistently understands complex, multi-faceted queries and identifies assessments that meet both explicit and implicit user needs.
- **Recommendation Balance**: The explicit instructions in the LLM prompt ensure a proper mix of technical and behavioral assessments is provided whenever a query requires it.
- **Final Performance**: The final iteration achieves an estimated **Mean Recall@10 in the 0.80 - 0.90 range**, demonstrating a highly accurate and reliable recommendation engine that significantly outperforms the baseline semantic search approach.

6. Tech Stack & Deployment Details

The system is built with a modern, scalable technology stack and is ready for production deployment.

- **Technology Stack**:
    - **Backend**: **FastAPI** with **Uvicorn** for a high-performance, asynchronous API.
    - **Frontend**: **Streamlit** for rapid development of an interactive and user-friendly web interface.
    - **AI / ML**:
        - **Groq API** (Llama 3.1 8B Instant) for fast LLM inference.
        - **Sentence Transformers** (all-MiniLM-L6-v2) for text embeddings.
        - **FAISS** for efficient vector similarity search.
    - **Data Processing**: **Pandas**, **BeautifulSoup**, and **Requests**.
- **Deployment Details**:
    - **API Deployment**: The FastAPI application is container-ready and can be deployed to cloud services such as **Render**, **Railway**, or Heroku. It is configured to run via the command: uvicorn main:app --host 0.0.0.0 --port $PORT.
    - **Frontend Deployment**: The Streamlit application is designed for easy deployment on **Streamlit Community Cloud**.
    - **Configuration**: The system requires a .env file at the root to store sensitive keys, primarily the GROQ_API_KEY. The API and frontend URLs need to be configured for communication in a deployed environment.