SHL Assessment Recommendation System- Technical Approach

1. Executive Summary

This project implements a **Hybrid RAG (Retrieval-Augmented Generation)** system to map natural language job descriptions to SHL assessments. The solution addresses the two core challenges of the assignment: **data accessibility** (scraping a client-side rendered catalog) and **recommendation balance** (ensuring a mix of hard and soft skill assessments).

2. High-Level Architecture

The system follows a three-stage pipeline: **Ingestion $\rightarrow$ Retrieval $\rightarrow$ Reasoning**.

3. Data Pipeline & Resilience Engineering

Challenge: The SHL product catalog uses React for dynamic rendering and infinite scrolling. During testing, the host exhibited severe rate-limiting (timeouts > 60s).

Solution:

- **Custom Playwright Scraper:** Built a browser-automation scraper (bypassing standard HTTP requests) to handle client-side rendering.
- **"Priority Scrape" Fallback:** To guarantee system reliability under network throttling, the scraper was engineered to prioritize high-value assessments identified in the training set (e.g., Python, Java, Sales) before attempting to fetch the tail-end of the catalog.
- **Data Integrity:** Implemented strict filtration to exclude "Pre-packaged Job Solutions" and enforce unique entries.

4. The Intelligence Layer (Context Engineering)

A standard vector search fails to capture the nuance of "balance." My solution uses a two-stage process to mimic a human recruiter's decision-making.

Stage A: Semantic Retrieval (Recall)

- **Model:** sentence-transformers/all-MiniLM-L6-v2
- **Method:** Dense vector embeddings are generated from product names, descriptions, and test types.

- **Operation:** For every query, the system performs a Cosine Similarity search to retrieve the **Top 25** candidates. This high-recall step ensures no relevant assessment is missed due to keyword mismatch.

Stage B: LLM Reranking (Precision & Balance)

- **Model:** Google Gemini 1.5 Flash
- **Logic:** The Top 25 candidates are passed to the LLM with a system prompt designed to enforce **"Recommendation Balance."**
    - *Rule 1:* If a query implies multiple domains (e.g., "Java Developer" + "Team Player"), the output MUST include both Technical and Behavioral assessments.
    - *Rule 2:* Deduplicate overlapping tests to maximize utility within the 10-item limit.

5. Evaluation Strategy

- **Metric:** Recall@10 (Primary).
- **Optimization:** Initial baseline (keyword search) yielded poor results (~0.4). Switching to Hybrid RAG allowed the system to correctly map abstract JD terms (e.g., "Analyst") to specific products ("Verify Calculation"), significantly improving the score on the Train Set.

6. Technical Stack

- **Backend:** Python 3.9, FastAPI (Async)
- **AI/ML:** Sentence-Transformers (Local), Google Gemini API
- **Infrastructure:** Playwright (Scraper), ngrok (Public Tunneling)