GenAI Task: Build an SHL Assessment Recommendation System

Why are we asking you to do this assessment?

We use this assessment to evaluate the following core skills:

1. **Problem-Solving**
   Ability to understand and synthesize the problem, decompose it into manageable components, and design a solution that is coherent, scalable, and meaningful.

2. **Programming Skills**
   Ability to write clean, effective, and reliable code to solve the problem. This includes appropriate use of AI-assisted tools to accelerate development without compromising correctness or understanding.

3. **Context Engineering**
   Ability to deeply understand the provided context, constraints, and data, and leverage them to produce solutions that are accurate, relevant, and well-aligned with the problem requirements.

Indicators of Unsuccessful Submissions

Based on our experience evaluating thousands of such assessments, candidates are unlikely to be successful if:

1. **Insufficient programming skills and experience**
   Core programming skills are not strong enough to implement a robust, maintainable, and extensible solution.

2. **Vibe-coding is great but not replacement for foundations**
   The solution relies on "vibe-coding" or trial-and-error without demonstrating a deep understanding of the problem, assumptions, and trade-offs.

3. **Limited validation/testing rigor**
   The solution is not tested across a sufficiently diverse set of inputs, edge cases, or realistic queries to demonstrate reliability and generalizability.

Brief overview of hiring process

Our hiring process consists of the following stages:

1. **Take-Home Assessment**
   Evaluation of problem-solving ability, Gen AI programming skills, and context engineering.

2. **First-Round Interview**
   We take a deep-dive on your submission and do a technical discussion on different aspects i.e. reasoning behind design choices, and core programming fundamentals including live problem-solving and adaptation of solutions under changing constraints.

3. **Second-Round Interview**
   Discussion of experience, projects, and previous problem-solving experience.

4. **Third-Round Interview**
   Technical/Non-Technical discussion with hiring manager for understanding role alignment and motivation.

Problem Overview

Hiring managers and recruiters often struggle to find the right assessments for the roles that they are hiring for. The current system relies on keyword searches and filters, making the process time-consuming and inefficient. Your task is to build an **intelligent recommendation system** that simplifies this process. Given a **natural language query or a job description (JD) text or an URL (containing a JD)**, your application should return a list of relevant SHL assessments.
You can take a look at the data sources that you are going to work with here,
https://www.shl.com/solutions/products/product-catalog/

Your Task

Design and develop a web application that:

1. Takes a given natural language query or job description text URL
2. Recommends **minimum 5 (maximum 10)** most relevant "individual test solutions" from *here* in the tabular format. Please note that you need to ignore "Pre-packaged Job Solutions" category from this link
3. Each recommendation needs to have at least the following attributes
   - Assessment name
   - URL (as given in SHL's catalog)

Datasets Given

1. The actual data over which you need to build the recommendation engine can be found: link
   a. You need to crawl the assessment catalogue from the SHL website to be able to build this solution.
   b. Make sure that there are at least 377 **Individual Test Solutions** after crawling to the given website.
   c. Build the recommendation system by leveraging this downloaded data.
2. Once you have build your first solution you can use train data provided to validate and iterate over it.
   a. Datasets can be found at: link
      i. Labelled Train set: This contains a set of 10 queries labeled by humans, most relevant assessments from the catalog. This can be used to iterate over your prompts, improve your pipeline etc.
   b. You can use the labeled train data to iterate and improve your solution.
3. Once you have finalized your solution, you need to generate prediction on the test set provided:
   a. Datasets can be found on the same url as above: link
   b. Unlabeled test set: This dataset contains a set of 9 queries – on which you have generate and submit predictions.

Submission Materials

You need to submit the following items using this form

- 3 URLs,
  1. API end point which can be queried using a query or piece of text and returns result in JSON (API Configurations mentioned below in Appendix 2)
  2. URL of the code on GitHub which we can see (**URL of the complete code - Including your experiments and evaluation**)
  3. URL of the web application frontend to test the application
- 2-page document outlining your approach on how you solved this problem. *Most importantly*,

document the efforts you put into optimizing the overall performance score, including initial results and how you improved them. Write this document as concisely as possible with appropriate information

- 1-csv file with 2 columns query and predictions (Format given in Appendix 3). The csv file should contain the predictions on the given unlabeled test set.

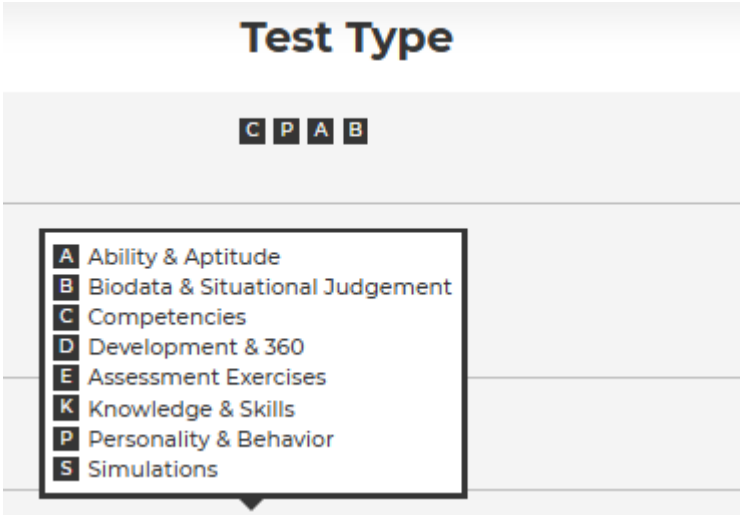
These points are vital for us to evaluate the technical completeness and robustness of your solution:

- **Ensure** that the API URL providing access to your recommendation engine is functional, your code is accessible via a public or private (shared with us) GitHub repository, and the csv file is in the correct format
- A .csv file in the prescribed format runs over the given test set (Format given in Appendix 3)

Evaluation Criteria (Core Logic)

We will be using the following criteria to evaluate your solution.

- **Solution Approach**
    - General Evaluation
        - We expect a clear and implementable strategy that directly addresses the problem statement. The code should reflect:
            - A well-defined pipeline connecting **data ingestion**, **retrieval**, and **recommendation**.
            - Logical flow between components (data → embedding → search → recommendation).
            - Modular, reproducible, and maintainable code structure.
            - Appropriate use of APIs, prompts, or models in line with the problem's requirements.
    - Data Pipeline
        - We expect:
            - Implementation of a data ingestion pipeline that scrapes or retrieves SHL product information.
            - Clean parsing and structuring of product data (titles, categories, URLs, etc.).
            - Use of efficient storage and retrieval mechanisms e.g., embeddings or vector databases
        - *Solutions built without scraping and storing SHL product catalogue from the website will be rejected*
    - Technology Stack & LLM Integration
        - We expect:
            - Use of **modern LLM-based or retrieval-augmented techniques** for query understanding or recommendation generation.
            - Justifiable use of the chosen frameworks (Langfuse, LangChain, LlamaIndex, etc.) based on functionality

        - *Solutions without clear LLM or retrieval-based integration will be rejected.*
    - Evaluation
        - We expect:
            - Implementation of **evaluation methods** to measure system accuracy and effectiveness.
            - Evaluation applied to the right stages (retrieval and final recommendation).
            - We have provided you with train data. This train data is given to help you evaluate and iterate over your solution.
        - *Solutions lacking measurable evaluation will be rejected.*
- **Performance and Relevance**
    - **Recommendation Accuracy:** The performance of assessment recommendation, measured by the Mean Recall@10 against the provided test set.
    - **Recommendation Balance:** The relevance and balance of recommended assessments based on query requirements.
        - **Requirement:** The system must intelligently balance recommendations when a query spans multiple domains. For instance, if a query pertains to both behavioral and technical skills, the results should contain a balanced mix of assessments.
        - **Example Scenario:**
            - **Sample Query:** "Need a Java developer who is good in collaborating with external teams and stakeholders."
            - **Expected Outcome:** The recommendation results should include a balanced set of assessments including hard and soft skills corresponding to both "Knowledge & Skills" (Test Type K) and "Personality & Behavior" (Test Type P) from the catalogue in this case. (Test Type is present in the catalogue)
            - 

Resources

You are not restricted to use these – feel free to use from anywhere else. Number of cloud platforms allow you to host applications and APIs for free for some time. Feel free to leverage those.

1. LLMs/Gemini Free APIs: *https://ai.google.dev/gemini-api/docs/pricing*
2. Cloud deployment platforms with free tier: https://github.com/cloudcommunity/Cloud-Free-Tier-Comparison

Index: Metrics to compute accuracy

Your solution will be assessed using the following **ranking evaluation metrics**:

1. Mean Recall@K

This metric measures how many of the **relevant assessments** were retrieved in the **top K recommendations**, averaged across all test queries.

$$Recall@K = \frac{\text{Number of relevant assessments in top K}}{\text{Total relevant assessments for the query}}$$

$$MeanRecall@K = \frac{1}{N}\sum_{i=1}^{N} Recall@K_i$$

where **N** is the total number of test queries.

Appendix 1: Sample Queries

Here are some of the queries that you can use to test your application. (Also present in the dataset given)

- I am hiring for Java developers who can also collaborate effectively with my business teams.
- Looking to hire mid-level professionals who are proficient in Python, SQL and Java Script.
- Here is a JD text, can you recommend some assessment that can help me screen applications. I am hiring for an analyst and wants applications to screen using Cognitive and personality tests

Appendix 2: API Structure & Endpoints

This section outlines the required API configuration for the take-home assignment. Your API must implement the endpoints described below exactly as specified to ensure proper evaluation of your submission.

Base requirements:

- Your API should be accessible via HTTP/HTTPS
- All responses should use proper HTTP status codes
- All data exchanges must be in JSON format

Required Endpoints:

1. **Health Check Endpoint:** This endpoint provides a simple status check to verify the API is running.

Request:

- Method: `GET`
- Path: `<YOUR-BASE-URL>/health`

Response:

- Content-Type: `application/json`
- Status Code: `200 OK` (if healthy)
- Body:

```
{
    "status": "healthy"
}
```

2. **Assessment Recommendation Endpoint:** This endpoint accepts a job description or Natural language query and returns recommended relevant assessments (At most 10, minimum 1) based on the input.

Request:

- Method: `POST`
- Path: `<YOUR-BASE-URL>/recommend`
- Content-Type: `application/json`
- Body:

```
{
    "query": "JD/query in string"
}
```

Response:

- Content-Type: `application/json`
- Status Code: `200 OK` (if successful)
- Body:

```
{
    "recommended_assessments": [
        {
            "url": "Valid URL in string",
            "adaptive_support": "Yes/No",
            "description": "Description in string",
            "duration": 60,
            "remote_support": "Yes/No",
            "test_type": ["List of string"]
        }
    ]
}
```

Response Fields Explanation: For the /recommend endpoint response ->

Field	Type	Description
url	String	Valid URL to the assessment resource
name	String	Name of the assessment
adaptive_support	String	Either "Yes" or "No" indicating if the assessment supports adaptive testing
description	String	Detailed description of the assessment
duration	Integer	Duration of the assessment in minutes
remote_support	String	Either "Yes" or "No" indicating if the assessment can be taken remotely
test_type	Array of Strings	Categories or types of the assessment

Example Response:

```
{
  "recommended_assessments": [
    {
      "url": "https://www.shl.com/solutions/products/product-catalog/view/python-new/",
      "name": "Python (New)",
      "adaptive_support": "No",
      "description": "Multi-choice test that measures the knowledge of Python programming, databases, modules and library. For...",
      "duration": 11,
      "remote_support": "Yes",
      "test_type": ["Knowledge & Skills"]
    },
    {
      "url": "https://www.shl.com/solutions/products/product-catalog/view/technology-professional-8-0-job-focused-assessment/",
      "name": "Technology Professional 8.0 Job Focused Assessment",
      "adaptive_support": "No",
      "description": "The Technology Job Focused Assessment assesses key behavioral attributes required for success in fast-paced roles.",
      "duration": 16,
      "remote_support": "Yes",
      "test_type": ["Competencies", "Personality & Behaviour"]
    }
  ]
}
```


Testing your API: Before submitting your API link, please verify that ->

1. Both endpoints are functioning correctly
2. The response formats match exactly what is given

Appendix 3: Submission Data format

We have an automated pipeline that will evaluate your results so make sure you follow the format as given.

- D

The CSV file should be submitted in the format below.

Query	Assessment_url
Query 1	Recommendation 1 (URL)
Query 1	Recommendation 2 (URL)
Query 1	Recommendation 3 (URL)
…	…
Query 2	Recommendation 1

For example,

Note the submission should be in exactly the above format, if the above format is not followed then you will not be scored.