**Project Overview**

The goal of this project is to build an intelligent recommendation system for SHL assessments that simplifies the process of finding relevant tests for hiring managers. Given a natural language query or job description, the system recommends up to 10 assessments from the SHL catalog, ensuring accuracy and relevance.

# **Approach**

1. Data Scraping and Representation :
   * The SHL assessment data was extracted using Deepseek AI and stored in a structured JSON format (**data.json**).
   * The dataset includes metadata such as assessment names, descriptions, categories, durations, and other attributes.
   * Preprocessing steps include combining relevant fields (e.g., name, description, category) into a single text column for efficient querying.
2. Query Understanding :
   * The Gemini API is used to extract key features from the input query, such as skills, duration, and test type.
   * This ensures that the system understands the intent behind the query and maps it to relevant assessments.
3. Recommendation Engine :
   * A TF-IDF vectorizer is used to preprocess the dataset and user queries.
   * Recommendations are generated using cosine similarity , which measures the similarity between the query and the assessments in the dataset.
   * The top 3–10 most relevant assessments are returned based on the similarity scores.
4. Frontend and Backend :
   * The frontend is built using Streamlit , providing an intuitive interface for users to input queries and view results in a tabular format.
   * The backend is implemented using FastAPI , exposing an endpoint for programmatic access to recommendations.
5. Evaluation :
   * The system is evaluated using Mean Recall@3 and MAP@3 metrics to measure the quality of recommendations.
   * Benchmark queries were tested against the system to ensure high accuracy and relevance.

#### **Tools and Libraries Used**

* Data Processing :
  + **pandas**: For loading and preprocessing the **data.json** file.
  + **scikit-learn**: For TF-IDF vectorization and cosine similarity calculations.
* Natural Language Processing :
  + Gemini API : For extracting key features from natural language queries.
* Web Development :
  + Streamlit : For building the web-based user interface.
  + FastAPI : For creating the API endpoint to handle programmatic requests.
  + **uvicorn**: To serve the FastAPI application.
* Environment Management :
  + **python-dotenv**: For securely managing environment variables like the Gemini API key.
* Deployment :
  + Render : For hosting FastAPI endpoint.
  + Streamlit cloud : For hosting the Streamlit app
  + GitHub : For version control and integration with deployment platforms.

**Key Features**

* User-Friendly Interface : Streamlit provides a clean and functional UI for entering queries and viewing recommendations.
* Scalable Backend : FastAPI ensures fast and reliable responses for both web and API-based interactions.
* Modular Design : Separate files (**app.py**, **api.py**, **utils.py**) ensure code maintainability and reusability

**Future Improvements**

1. Fine-tune the Gemini API prompts for better query understanding.
2. Experiment with advanced similarity metrics like BM25 for improved recommendations.
3. Expand the dataset to include additional metadata or assessments.

**Conclusion**

This project successfully demonstrates an end-to-end solution for recommending SHL assessments based on natural language queries. By leveraging modern tools like Gemini API, Streamlit, and FastAPI, the system achieves high accuracy and usability while adhering to best practices in software development and deployment.