**GenAI Assessment Recommendation Tool: Approach**

**Problem:** To develop a user-friendly tool that leverages GenAI to recommend relevant SHL assessment solutions based on natural language queries, job descriptions, or URLs, using a product catalog.

**Approach:**

1. **Data Acquisition and Preparation:**
   * A catalog.csv file containing SHL assessment metadata (name, description, skills, job roles, duration, ID, remote testing, adaptive IRT, test type, URL) was used as the knowledge base.
   * The catalog was loaded and validated using pandas in recommend.py, ensuring all required columns and data types were present.
   * A text\_blob column was created by concatenating relevant text fields (name, description, skills, job roles) to serve as the basis for semantic similarity calculations.
2. **Semantic Embedding Generation:**
   * The sentence-transformers library and the pre-trained all-MiniLM-L6-v2 model were employed to generate dense vector embeddings for each assessment's text\_blob. This model provides a balance between efficiency and semantic understanding.
   * Embeddings were generated and cached using streamlit.cache\_data for performance.
3. **Recommendation Logic:**
   * The core recommendation function get\_top\_k in recommend.py takes a user query (or extracted text from a job description/URL) as input.
   * Cosine similarity between the query embedding and the assessment embeddings is calculated using util.pytorch\_cos\_sim.
   * Assessments are ranked based on their similarity scores.
   * Results are filtered based on a minimum similarity threshold (MIN\_SCORE\_THRESHOLD) and optional maximum duration.
   * The top-k matching assessments are returned as a list of dictionaries, including relevant metadata and a similarity score.
4. **User Interface (Web Application):**
   * Streamlit (streamlit) was used to build an interactive web application (app.py).
   * The UI allows users to input queries, job descriptions (via text area or URL), or upload a custom catalog.
   * Job descriptions from URLs are fetched and parsed using requests and BeautifulSoup4.
   * Keywords are optionally extracted from the input using spacy for enhanced understanding (though not directly used in the core similarity calculation).
   * Recommendations are displayed in a user-friendly format (e.g., table).
   * A conceptual API endpoint structure is documented in the sidebar.
5. **Deployment:**
   * The application is designed to be deployed on Streamlit Cloud, leveraging a requirements.txt file to manage dependencies.

**Key Technologies:**

Python, Streamlit, pandas, sentence-transformers, torch, requests, BeautifulSoup4, spacy.

**Conceptual API:**

A potential REST API (POST /recommendations) would accept a "query" in JSON and return JSON results of recommended assessments with metadata and match scores.

**Outcome:**

The resulting tool enables users to easily discover relevant SHL assessment solutions through a natural language interface, leveraging the power of semantic similarity to bridge the gap between job requirements and assessment capabilities. The web-based interface makes the tool accessible and user-friendly.