Project Name: Assessment Recommendation Engine – Personalized Assessment Suggestions

---

Problem Statement:
Students often struggle to find relevant assessments or tests that match their goals. This tool recommends the most suitable assessments based on user input, making selection faster and personalized.

Objectives:

- Match users with suitable assessments based on their goals.
- Process over 1,000 assessment listings efficiently.
- Provide recommendations quickly via a responsive interface.
- Maintain a modular and scalable architecture for easy updates.

Input & Output:

- **Input:** A user-defined goal or skill (text query).
- **Output:** A ranked list of recommended assessments based on similarity.

Technologies Used:

- Python for overall development.
- **NLP Techniques:** TF-IDF (Term Frequency–Inverse Document Frequency) for text representation.
- **Similarity Measure:** Cosine Similarity to compare user query and assessment content.
- **FastAPI** for backend API development.
- **Streamlit** for building an interactive front-end.
- **Modular Design:** Clear separation between feature generation, similarity logic, API, and UI.

System Architecture Workflow:

1. User inputs their goal via the Streamlit UI.
2. FastAPI backend receives the request.
3. TF-IDF processes both the user input and assessment descriptions into numerical vectors.
4. Cosine Similarity computes similarity scores.
5. The system returns the top assessment recommendations, displayed in the UI.

Modeling & Design Reasoning:

- **TF-IDF** was chosen because it effectively highlights keywords relevant to each assessment.
- **Cosine Similarity** was used for its efficiency in measuring text similarity in high-dimensional vector space.
- **FastAPI + Streamlit** combination was chosen to provide both a developer-friendly backend and an intuitive UI.

Performance & Scalability:

- Precomputed TF-IDF matrix for assessment listings ensures fast similarity computation at runtime.
- Modular design allows easy updates to the model or assessment list without massive code changes.
- Lightweight layout enables near real-time recommendations (<1 second per query).

Results:

- Engine successfully handles over 150 user interactions in testing.
- Users receive recommendations within 1 second of submitting a query.
- High user satisfaction (based on user tests) for recommendation relevance.

Deployment & Run Instructions:

1. Clone the repository.
2. Set up your Python environment (e.g., `pip install -r requirements.txt`).
3. Run the API server using `uvicorn main:app --reload`.
4. Launch the frontend via `streamlit run frontend.py`.
5. Enter your goal in the UI to get matching assessments instantly.

Future Improvements:

- Add a feedback loop to learn from accepted vs rejected recommendations.
- Integrate embeddings (e.g., Word2Vec or Sentence-BERT) for better semantic matching.
- Containerize the application for robust deployment.
- Add logging & analytics to monitor usage patterns.