**UIUC Student Recommender System**

**1. Data Collection and Preprocessing**

Before fine-tuning, you need structured data from:

* **UIUC Course Catalog & Historical GPAs** (e.g., GPA Analyzer, UIUC Course Explorer, Reddit discussions, and past grade distributions)
* **Curriculum Maps & Major Requirements** (CS, Engineering, Business, etc.)
* **Course Reviews and Instructor Feedback** (ICES scores, student surveys)
* **RSO Recommendations** (RSO database, UIUC Involvement Fair data, student testimonies)
* **Past Course Selections & Patterns** (potentially scraped from public student discussions)

You can preprocess this into structured datasets with:

* **Tabular Data** (course → GPA distributions, instructors, major relevance)
* **Graph Data** (network of courses & prerequisites)
* **Textual Data** (student experiences, advice)

**2. Retrieval-Augmented Generation (RAG)**

Instead of fully fine-tuning a model (which can be expensive and require continuous updates), **use RAG to fetch the most relevant data** before generating responses:

1. **Build a Vector Database (FAISS, Weaviate, Pinecone)** to store past course reviews, GPA distributions, and RSO descriptions.
2. **Chunk & Embed Data**: Convert curriculum maps, historical data, and student feedback into embeddings for efficient retrieval.
3. **Augment Model Inputs**: Before responding, the LLM queries relevant information from the database, ensuring factual accuracy.

Example:

* **User asks:** “What’s a good CS technical elective for AI?”
* **LLM retrieves:** Past student GPAs, reviews, and course relevance from the vector DB.
* **LLM responds:** With a personalized, data-backed suggestion.

**3. Adaptive Recommendation Algorithm**

To personalize recommendations, you can implement:

**Collaborative Filtering**

* If you have access to **student course history** (e.g., a dataset of courses taken per major), use **matrix factorization**(SVD, ALS) or **neural collaborative filtering** (NCF) to recommend courses based on what similar students took.
* Example:
  + A student asks for a **Business minor elective**.
  + The system finds **similar students’ choices** and suggests courses with high satisfaction.

**Content-Based Filtering**

* If you lack direct course history, use **text-based similarity**:
  + Encode courses with TF-IDF / BERT embeddings.
  + Rank them based on cosine similarity.
  + Example: If a student likes **CS 441 Applied Machine Learning**, suggest **CS 446 Machine Learning**.

**Hybrid Approach**

* Combine **collaborative filtering + content similarity** for better recommendations.

**4. Fine-Tuning**

Once your RAG pipeline is strong, you can **fine-tune an LLM on structured queries**:

1. **Prepare Supervised Data** (QA pairs with real student recommendations)
2. **Train on Course & RSO Descriptions** (so it understands UIUC-specific terms)
3. **Fine-Tune on Chat Logs** (to improve conversational quality)

**Model Choice:**

* **Smaller Open-Source LLMs (Mistral 7B, Llama 3, Mixtral)** for cost-efficient fine-tuning.
* **LoRA (Low-Rank Adaptation)** for efficient fine-tuning without retraining the full model.

**5. UI / Chatbot Integration**

* Deploy via **FastAPI + LangChain** to provide an interactive search experience.
* Create a **web app or Discord bot** where students can ask for recommendations.