Stack A — “University IT-friendly” (Postgres-first, easy to host)

Best suited for: those who want a solution that resembles a typical campus web app (Postgres, Docker), with minimal new infrastructure requirements.

- **Zotero → ingestion**
  - Zotero desktop + **Better BibTeX “Keep updated” auto-export** to JSON/BibTeX/CSL JSON (easy, reliable sync)
- **Processing**
  - Python: pydantic, pandas
  - PDF/text extraction: pymupdf (or unstructured if you need more layout handling)
- **Embeddings**
  - OpenAI text-embedding-3-large (strong quality) or text-embedding-3-small (cheaper)
- **Vector store**
  - **Postgres + pgvector** (keeps everything in one DB; easier approvals/backups)
- **RAG framework**
  - **LlamaIndex** (great “documents-first” indexing, metadata filters, citations)
- **App**
  - FastAPI backend + Next.js (or simple React) frontend
  - Auth: campus SSO (SAML/OIDC) if needed; otherwise basic auth to start
- **Why it fits your use case**
  - You’ll want strong **metadata filtering** (discipline, grade level, activity type, learning outcome, “policy vs pedagogy,” etc.), and pgvector + LlamaIndex handles that cleanly.

Stack B — “All-local / offline-capable” (no external APIs required)

Best suited for: privacy/FERPA concerns, or when you want to run fully on a local machine or lab server.

- **Zotero → ingestion**
  - Better BibTeX export (auto-updating)
  - Optional: Better BibTeX **JSON-RPC** if you want programmatic exports/automation
- **Embeddings (local)**
  - SentenceTransformers (e.g., BGE / E5-family) via sentence-transformers
- **Vector store**
  - **Qdrant** (excellent open-source option; good filtering; easy Docker)
  - (Simpler fallback for prototyping: Chroma)
- **RAG framework**
  - LlamaIndex *or* Haystack (Haystack is very “pipeline/production” oriented)
- **Local LLM**
  - Ollama or vLLM (depending on your hardware)
- **App**
  - FastAPI + simple UI (Streamlit is fine for an MVP)

- **Why it fits your use case**
  - Let's you keep **all papers + prompts + outputs on-prem**, which universities often prefer.

Stack C — "Managed cloud / fastest to scale" (best UX, minimal ops)

Best if: you want the smoothest production experience and don't want to run your own vector DB.

- **Zotero → ingestion**
  - Better BibTeX auto-export (or Zotero API if you're using group libraries heavily)
- **Embeddings**
  - OpenAI text-embedding-3-large/small
- **Vector store (managed)**
  - Pinecone / Weaviate Cloud / MongoDB Atlas Vector Search (pick based on what your team already uses)
- **RAG framework**
  - LangChain (fast iteration + lots of integrations) or LlamaIndex (strong indexing + retrieval abstractions)
- **Reranking (high impact for research papers)**
  - Add a reranker step (Cohere rerank or a local cross-encoder) to improve "right paper, right chunk" retrieval
- **App**
  - Next.js UI + FastAPI (or serverless functions)
- **Why it fits your use case**
  - Great when you expect lots of faculty users and want reliability + fast search at scale.