ScholarSync

Fase 1: Desain Algoritma & Persiapan Data (Minggu 1-2)

Tujuan: Memahami logika di balik layar dan menyiapkan data.

1.1. Setup Environment

- Instalasi Python (gunakan virtual environment: `venv` atau `conda`).
- Instalasi library kunci: `langchain`, `langchain-experimental` (untuk semantic chunker), `qdrant-client` (atau Pinecone), `openai` (untuk embedding/LLM), `rank_bm25` (untuk sparse search), `streamlit`.
- Siapkan API Keys: OpenAI API, Cohere API (untuk Reranking), Qdrant/Pinecone API.

1.2. Implementasi Semantic Chunking

- **Logika:** Jangan potong teks per 500 kata. Gunakan *Semantic Chunking*.
- **Algoritma:**
  1. Ubah kalimat menjadi embedding.
  2. Hitung *cosine distance* antar kalimat berurutan.
  3. Jika jaraknya besar (berarti topik berubah), lakukan *split* (pemotongan) di situ.
- **Output Fase Ini:** Script Python yang bisa membaca 1 PDF dan menghasilkan list potongan teks (chunks) yang "masuk akal" secara makna.

1.3. Desain Hybrid Search Strategy

- Tentukan bobot *Alpha*. Rumus umum: `Hybrid_Score = (Alpha * Dense_Score) + ((1 - Alpha) * Sparse_Score)`.
- Biasanya `Alpha = 0.5` atau `0.7` (lebih mengutamakan makna daripada keyword persis).

Fase 2: Membangun Backend Pipeline (Minggu 3-4)

Tujuan: Membuat mesin RAG yang bisa mencari dan memfilter informasi.

2.1. Vector Database Ingestion (Pemasukan Data)

- Pilih Database: Qdrant (disarankan karena support hybrid search native dan cepat) atau Pinecone Serverless.
- **Dense Vector:** Gunakan `text-embedding-3-small` (OpenAI) untuk mengubah text chunk menjadi angka.
- **Sparse Vector:** Gunakan algoritma BM25 untuk membuat vector keyword.
- Upload kedua jenis vector ini ke database.

2.2. Implementasi Retrieval (Pencarian Awal)

- Buat fungsi Python `retrieve_documents(query)` yang melakukan pencarian ke Vector DB.
- Target: Ambil Top-20 atau Top-30 dokumen yang relevan.

2.3. Implementasi Reranking (Penyaringan Cerdas)

- Integrasikan **Cohere Rerank API**.
- **Input:** Query User + Top-30 dokumen dari langkah 2.2.
- **Proses:** Cohere akan membaca ulang dan memberi skor relevansi baru.
- **Output:** Ambil Top-5 dokumen terbaik (Golden Chunks).

Fase 3: Integrasi LLM & Logic Generasi (Minggu 5)

Tujuan: Membuat AI bisa menjawab berdasarkan dokumen yang ditemukan.

3.1. Prompt Engineering

- Buat template prompt sistem yang ketat.
- Contoh:

> "Kamu adalah asisten peneliti. Jawab pertanyaan HANYA berdasarkan konteks berikut: {context}. Jika tidak ada di konteks, katakan tidak tahu. Sertakan referensi di akhir jawaban."

3.2. Generation Chain

- Gabungkan pipeline: `Query` → `Hybrid Search` → `Reranking` → `Prompt` → `LLM (GPT-3.5-turbo/GPT-4o)` → `Jawaban`.

Fase 4: Frontend & User Interface (Minggu 6)

Tujuan: Membuat antarmuka yang bisa dipakai pengguna.

4.1. Membangun UI Streamlit

- **Sidebar:** Widget `st.file_uploader` untuk upload PDF jurnal. Tombol "Process Documents".
- **Main Area:** Interface chat (`st.chat_message`) untuk tanya jawab.
- **Expander:** Fitur `st.expander` bertuliskan "Lihat Sumber Referensi" untuk menampilkan potongan teks asli yang dipakai AI (untuk validasi).

4.2. Menghubungkan UI ke Backend

- Saat user klik "Process", jalankan script ingestion (Fase 2.1).
- Saat user kirim chat, jalankan generation chain (Fase 3.2).

Fase 5: Deployment & Finalisasi (Minggu 7-8)

Tujuan: Online-kan aplikasi agar bisa diakses dosen/penguji.

Ingat strategi Backend (Vercel) + Frontend (Streamlit Cloud) yang kita bahas sebelumnya.

5.1. Refactoring ke API (FastAPI)

- Pindahkan semua logic berat (Ingestion, Search, Rerank) ke dalam file `api.py` menggunakan **FastAPI**.
- Buat endpoint:
  - `POST /upload`: Menerima file PDF, lakukan chunking & upload ke DB.
  - `POST /chat`: Menerima pertanyaan, balikan jawaban + sumber.

5.2. Deploy Backend ke Vercel

- Buat file `vercel.json` untuk konfigurasi Python serverless function.
- Push code backend ke GitHub → Connect ke Vercel.

5.3. Deploy Frontend ke Streamlit Community Cloud

- Ubah kode Streamlit agar tidak memanggil fungsi lokal, tapi melakukan `requests.post()` ke URL API Vercel kamu.
- Push code frontend ke GitHub (repo terpisah atau folder terpisah) → Connect ke Streamlit Cloud.

5.4. Testing & Dokumentasi

- Uji coba dengan jurnal yang rumit. Apakah jawaban akurat?
- Tulis dokumentasi di README GitHub (Sangat penting untuk nilai proyek akhir). Jelaskan arsitektur Hybrid + Reranking + Semantic Chunking.