**Plan (cleaning-first, retrieval-second)**

**1) Scope and rules**

* **Only public pages under https://liftoff.io/resources/ (blogs and case studies).**
* **Respect robots.txt, add a polite User-Agent, throttle requests, and back off on 429/5xx.**
* **Don’t hit gated or parameterized endpoints beyond the resources filters/pagination.**

**2) URL discovery**

* **Start from listing pages:**
  + **Blogs: resources/?type=post**
  + **Case studies: resources/?type=case-study (or whatever the site uses for case studies; the discovery script will follow pagination links and category filters automatically).**
* **Follow pagination and collect individual article links.**
* **Normalize URLs (strip fragments, resolve relative paths), dedupe by canonical <link rel="canonical"> when available.**

**3) Fetch + clean (80% effort here)**

* **Fetch pages with retries, jitter, and caching to disk.**
* **Extract main content with trafilatura (fallback: readability-lxml).**
* **Normalize:**
  + **Fix unicode (ftfy), collapse whitespace, remove boilerplate (menus, footers), strip “share” blocks.**
  + **Merge broken sentences across headings/page breaks.**
  + **Keep structure: title, author (if present), date, section headings H2/H3.**
* **Validate language (skip non-English or mark it).**
* **Produce chunkable clean text + rich metadata:**
  + **url, canonical\_url, title, type (post | case-study), published\_date (parsed), last\_seen, section\_titles, word\_count, hash.**
* **Save both raw HTML and clean JSONL for traceability.**

**4) Chunking + embeddings**

* **Chunk at ~500 tokens with 80-token overlap using semantic-aware splitter (paragraph + sentence boundaries first, then by length).**
* **Add metadata to each chunk: doc\_id, url, title, type, section\_title, published\_date.**
* **Embeddings: sentence-transformers/all-MiniLM-L6-v2 (fast, strong baseline).**

**5) Indexing in Chroma**

* **One collection: liftoff\_resources.**
* **Persist to ./index/.**
* **Store full metadata per chunk for precise filtering.**

**6) Retrieval strategy**

* **Default: vector search top\_k=8.**
* **Re-rank pass (optional): lightweight BM25 over the retrieved candidates using their texts.**
* **Filters:**
  + **If query contains “case study”, prefer type=case-study.**
  + **If query mentions a year or product, prefer matching published\_date / keywords.**
* **Confidence heuristic: average top-5 cosine; if < threshold, expand search (k=15) and retry.**

**7) LLM answer (Grok)**

* **Prompt template: short, strict, grounded. Ask for attributed answer with inline citations like [title, year].**
* **Output: final answer + list of source URLs used.**

**8) Ops + safety**

* **Rate limit: 1–2 req/sec; exponential backoff.**
* **Cache fetched pages to avoid re-hits during development.**
* **Idempotent ingest: skip unchanged pages via content hash.**
* **Simple eval: a few canned queries to spot-check retrieval and citation quality.**

**Minimal repo (≤15 files)**

**perl**

**CopyEdit**

**liftoff-rag/**

**├─ README.md**

**├─ requirements.txt**

**├─ .env.example**

**├─ seeds.json # optional: starting listing URLs and type labels**

**├─ app.py # FastAPI or Streamlit: /ingest and /rag/query**

**├─ src/**

**│ ├─ discover\_urls.py # crawl listing pages, follow pagination, write urls.jsonl**

**│ ├─ fetch\_and\_clean.py # download pages, extract main content, write clean.jsonl + raw/**

**│ ├─ chunk\_and\_embed.py # chunk clean docs, embed, upsert into Chroma**

**│ ├─ rag\_query.py # retrieval + optional rerank + Grok call**

**│ └─ utils\_clean.py # text normalization, date parsing, boilerplate removal**

**├─ data/**

**│ ├─ raw/ # saved html snapshots (files)**

**│ └─ clean.jsonl # one record per document with clean text + metadata**

**└─ index/ # Chroma persist dir**

**That’s 10 files plus three directories.**

**What each file does (brief)**

* **README.md — how to run ingestion, index, and query; env setup; legal/robots notes.**
* **requirements.txt — httpx[http2], trafilatura, readability-lxml, beautifulsoup4, ftfy, python-slugify, dateparser, sentence-transformers, chromadb, langchain, langgraph, fastapi or streamlit, uvicorn.**
* **.env.example — GROK\_API\_KEY=, USER\_AGENT=YourBot/0.1, RATE\_LIMIT\_RPS=1.5.**
* **seeds.json — listing endpoints and labels (e.g., { "resources": [{"url":"https://liftoff.io/resources/?type=post","type":"post"},{"url":"https://liftoff.io/resources/?type=case-study","type":"case-study"}] }).**
* **app.py — two endpoints or pages:**
  + **POST /ingest → runs discover → fetch\_clean → chunk\_embed.**
  + **POST /rag/query → runs retrieval + Grok and returns answer + sources.**
* **src/discover\_urls.py — fetch listing pages, follow pagination, extract article links, output urls.jsonl.**
* **src/fetch\_and\_clean.py — fetch each URL; save data/raw/{slug}.html; extract to structured doc; append to data/clean.jsonl.**
* **src/chunk\_and\_embed.py — read clean.jsonl, chunk, embed, upsert to Chroma collection.**
* **src/rag\_query.py — compose retriever (vector + optional BM25 rerank), build prompt, call Grok, return structured result.**
* **src/utils\_clean.py — helpers: unicode fix, boilerplate removal rules, canonical URL, date parsing, section title stitching.**

**Run order**

1. **Discover**

**css**

**CopyEdit**

**python -m src.discover\_urls**

1. **Fetch + Clean**

**css**

**CopyEdit**

**python -m src.fetch\_and\_clean**

1. **Chunk + Embed**

**css**

**CopyEdit**

**python -m src.chunk\_and\_embed**

1. **Query**

* **API: uvicorn app:app --reload then POST /rag/query with { "query": "..." }**
* **or Streamlit: streamlit run app.py**

**Cleaning details that score points**

* **Strip common blocks: nav, footer, “share”, newsletter CTAs, author bio boxes.**
* **Merge short adjacent paragraphs if both < 50 chars to avoid orphaned fragments.**
* **Preserve headings as section\_title metadata; they’re gold for relevance.**
* **Parse and normalize dates to ISO; some listings use “Updated …”.**
* **Store word\_count and a min threshold (skip tiny stubs).**
* **Keep hash of the cleaned main content; skip re-embed if unchanged.**

**Retrieval tweaks that impress without bloat**

* **Intent hinting: if the query mentions “performance”, “CPI”, “ROAS”, boost chunks with those tokens.**
* **Type filter: if query says “case study”, set where={"type":"case-study"} in Chroma.**
* **Temporal nudge: if query has a year, filter or sort by published\_date.**
* **Safety net: if average cosine < 0.25, increase k and broaden filters, then retry once.**

**If you want, I can fill in stubs for each script so you can drop in your keys and run end-to-end in an hour.**