Technical Challenge: Policy Discovery, Extraction, and LLM-Driven Structuring

Goal

Build an end-to-end system (Python + PHP) that discovers legal policy pages, extracts structured information using an LLM, and exposes the results via a minimal PHP chat interface.

The solution **must be compatible with the n8n AI Starter Kit** (n8n, Ollama, Qdrant, PostgreSQL).

Provided Files

List 1 (List 1.csv)

Each row represents a company/domain to process.

Columns

- id – unique identifier (must be preserved)
- name – company/service name
- domain – primary domain (entry point)
- generic_email – may be empty
- contact_email – may be empty
- privacy_email – may be empty
- delete_link – may be empty
- country – may be empty

Many fields are intentionally **missing or null**.

Template 1 (Template 1.xlsx)

A **policy-scope matrix**:

- Column SCOPES defines standardized privacy/terms categories (e.g. *Registration*, *Legal and Security Purposes*, *Customization*).
- Columns Company #1, Company #2, etc. indicate whether a scope applies (x).

Your task

- Convert this template into a **normalized SQL table** where:
    - each row represents **one domain/company**
    - each scope becomes a structured boolean or categorical field
- The populated table must be derived **only from the scraped policy text via LLM extraction**.

Core Objectives (Required)

Objective 1 — Policy Page Discovery

For each domain in List 1:

- Automatically discover:
    - Privacy Policy page
    - Terms & Conditions / Terms of Use page
- Use crawling + heuristics (footer links, sitemap, link text).
- Output canonical URLs with basic discovery metadata.

Objective 2 — Scraping and Vector Storage

- Scrape discovered policy pages.
- Clean and chunk text.
- Store chunks in a **vector database** (recommended: Qdrant).
- Each vector record must include metadata:
    - id (from List 1)
    - domain
    - doc_type (privacy | terms)
    - source_url

Objective 3 — LLM Parsing into SQL (Template 1)

- Use an LLM to:
    - retrieve relevant chunks from the vector DB
    - determine which **Template 1 scopes apply**
- Populate a PostgreSQL table that:
    - matches the Template 1 scope structure
    - is keyed by id or domain
- Output must be validated (structured JSON → SQL insert/upsert).

Objective 4 — PHP Chat Interface

- Build a **simple PHP web page** with:
    - a chat window
    - user questions answered by an LLM
- The LLM must:
    - query the vector DB (policy text)
    - query the SQL DB (structured scope data)
- Responses must cite the relevant policy URL(s).

Bonus Objective (Important)

Maximize enrichment of missing List 1 fields using scraped data and LLM reasoning:

Attempt to extract and populate:

- generic_email
- contact_email
- privacy_email
- delete_link (account/data deletion URL)
- country (inferred from legal text, address, or jurisdiction)

Store:

- extracted value
- source URL
- confidence or reasoning (free-text or JSON)

Bonus scoring strongly favors:

- correctness over completeness
- traceable justification

Technical Expectations

- Python: crawling, scraping, vector ingestion, LLM extraction
- PostgreSQL: structured data storage (Template 1 + enriched List 1 fields)
- PHP: minimal UI only (logic may live in Python or n8n)
- n8n (recommended): orchestration, scheduled runs, chat webhook

---

Deliverables

1. Source repository with README
2. Docker Compose (or n8n starter-kit integration)
3. SQL schema + migration scripts
4. Example output for at least 3 domains
5. Short engineering notes (design tradeoffs, limitations)

Evaluation Focus

- Correct mapping of real policy text → Template 1 scopes
- Reliable discovery of policy pages
- Quality of enrichment of missing List 1 fields
- End-to-end functionality and reproducibility