**Stack (keep it shippable in 48h)**

* **API & Orchestrator:** FastAPI + **LangGraph** (for the agent pipeline/state)
* **LLM provider:** OpenAI (gpt-4o-mini) or local **Ollama** (llama-3-8b) behind a small adapter
* **Vector store (laws & snippets):** **Chroma** (local, persistent)
* **Primary DB (runs, comments, artifacts):** **SQLite** (swap to Postgres later via env)
* **Background jobs:** FastAPI BackgroundTasks (Day 1), optional **Redis+RQ** (Day 2)
* **Scraping:** Bing/SerpAPI search API + **trafilatura** / **Playwright** (only if a page needs JS)
* **Evidence generation:** HTML template → **WeasyPrint** or wkhtmltopdf → PDF
* **Storage:** local ./uploads and ./evidence (S3 optional)
* **Schema & validation:** **Pydantic v2** everywhere (LLM JSON is untrusted until validated)

**Shared data model (no code, just the shape)**

* **features**: id, title, description, regions, created\_at
* **documents**: id, feature\_id, filename, sha256, path
* **runs**: id, feature\_id, status{queued|running|done|error}, decision, confidence, risk, result\_json, created\_at
* **findings**: run\_id, regulation\_id/url, obligation, citation\_snippet
* **comments**: feature\_id/run\_id, vote{+1|-1}, tag, note
* **agent\_weights**: agent\_name → weight (start at 1.0, 0.7–1.3 range)
* **retrieval\_index**: index\_id, created\_at, domain\_allowlist, source\_list

**Agents (mapped to your idea)**

1. **Agent-LawFinder (A1)**
   * Inputs: feature summary, regions
   * Actions: web search → crawl → clean → chunk → embed → store in Chroma (index\_id)
   * Guardrails: domain allowlist (.gov, .eu, state legislature), date filters, dedupe
   * Output: { index\_id, sources:[{url,title,jurisdiction,snippet}] }
   * Fallback: if search fails, load curated pack (DSA, CA minors, FL minors, UT minors, NCMEC)
2. **Agent-Normalizer (A2)**
   * Inputs: title, description, parsed PRD/TRD text, regions
   * LLM task: convert free-text to **strict JSON** (actions, data\_types, user\_groups, triggers, risk\_factors)
   * Validation: Pydantic schema; if invalid → return {ok:false, errors}
3. **Agent-Governor (A3)**
   * Inputs: A1 + A2
   * Sub-checks:
     + **RuleChecker:** deterministic rules (e.g., minors → age\_gate; location + storage → data\_localization)
     + **RAGChecker:** retrieve top law chunks from index\_id, ask LLM “legal obligation vs. business rollout?” (JSON-only)
     + **RiskScorer:** simple formula (obligation weights + data sensitivity)
   * Aggregation: weighted majority (agent\_weights from feedback) → **Yes/No/Unclear**, confidence, obligations, regs, citations, clarifying questions
4. **Agent-Exporter (A4)**
   * Inputs: A3 result + inputs metadata
   * Output: PDF evidence (decision, reasons, citations), CSV row, ZIP bundle
   * Auto-download: API responds with Content-Disposition: attachment
5. **Agent-Converse (A5)**
   * Purpose: user chat; if A3 is **Unclear**, ask 1–2 clarifying questions; on reply, re-run A1–A3

**Day 1 — “MVP end-to-end” (front-to-back in one day)**

**Morning — environment & contracts**

* Create FastAPI app structure; add health endpoint.
* Define **Pydantic models** for: Feature intake, A1/A2/A3 outputs, Evidence metadata.
* Set up **Chroma DB** (persistent folder) and **SQLite** with tables above.
* Create a **domain allowlist** for A1 (europa.eu, legislation.gov.uk, ca.gov, flsenate.gov, le.utah.gov, ncmec.org, etc.).

**Midday — Agent A1 (search & index)**

* Hook a web search API (Bing/SerpAPI). Build queries from regions + keywords in the description.
* Crawl top results, clean with trafilatura, chunk 400–800 tokens (80 overlap), embed, persist to **Chroma**; collect {url,title,jurisdiction} for audit.
* Return {index\_id, sources}; if < N quality sources, also add curated pack docs.

**Afternoon — Agent A2 (normalize)**

* Prompt LLM to output **strict JSON** for actions, data\_types, user\_groups, triggers, risk\_factors.
* Validate with Pydantic; if invalid → produce validation errors and mark feature as needing clarifications.

**Late afternoon — Agent A3 (governance decision)**

* Implement **RuleChecker** (deterministic) and **RAGChecker** (retrieve from index\_id → JSON verdict with citations).
* Combine with **RiskScorer**; aggregate with \*\*agent\_weights` (start all 1.0).
* Persist to **runs** with decision, risk, citations.
* Expose endpoints:
  + POST /ingest (save docs)
  + POST /analyze (run A1→A3 synchronously for MVP; return result JSON)
  + GET /runs/{run\_id} (read)

**Evening — Frontend integration & sanity**

* Point your **Analyze** button to POST /analyze; display returned result.
* Manual tests:
  + Minor-focused feature → decision **Yes** with age\_gate and CA/FL/UT regs.
  + “Market testing in US” → **No** (business, not legal).
  + “Global except KR” → **Unclear** with 1–2 clarifying questions.

**Done =** you can upload, analyze, and see reasons/citations end-to-end.

**Day 2 — “Feedback, evidence, and resiliency”**

**Morning — Feedback->learning loop**

* Endpoint POST /feedback (vote, tag, note).
* **Update rule:** for the agent involved in a wrong or right call, nudge weight:
  + Upvote → weight = min(1.3, weight + 0.05)
  + Downvote → weight = max(0.7, weight - 0.05)
* Maintain a **prompt memory** file per regulation: recurring tags (e.g., “Missing age-gate”) append a short “Do/Don’t” hint.
* A3 reads updated weights + memory each run.
* UI: your Feedback form already calls the endpoint; show “Thanks — learning applied”.

**Midday — Evidence & downloads**

* Build an HTML evidence template (feature meta, decision, obligations, citations, risk, agent votes, timestamps, run id, SHA256 of inputs).
* Render to PDF (WeasyPrint or wkhtmltopdf) → store under ./evidence/<run\_id>.pdf.
* Endpoints:
  + /export/evidence?run\_id=... → PDF
  + /export/csv?feature\_id=... → rows per run
  + /export/zip?run\_id=... → PDF + JSON result + CSV
* Make sure API returns with the **attachment header** so your frontend triggers download.

**Afternoon — Robustness & async**

* Wrap A1→A3 in a **BackgroundTask** so your API returns {run\_id, status:"queued"} quickly and the UI polls GET /runs/{id}.
* Optional upgrade: add **Redis+RQ** and one worker process if you expect multiple simultaneous analyses.

**Late afternoon — Observability & acceptance**

* Log per-agent JSON lines (inputs → outputs) with a correlation id (run\_id).
* Add two metrics: “citation validity %” and “% Unclear with clarifying Qs”.
* Acceptance checks:
  + 5 sample features produce correct Yes/No/Unclear.
  + PDF downloads and opens; citations link to sources; CSV imports in Excel.
  + Feedback up/down changes an agent weight (verify in DB) and nudges the next similar run.

**How everything fits your UI (no code changes required)**

* **Analyze:** calls /analyze → returns run\_id; your page polls /runs/{id} until **done** and fills Decision card.
* **Feedback:** calls /feedback; your existing “reset to original view” behavior is perfect.
* **Generate Evidence / Export CSV:** call /export/\*; file downloads automatically.
* **Audit Log:** surface GET /runs?feature\_id=... to fill your table.

**Practical tips**

* Start **online-first** (A1 web search). Keep the curated 5-reg pack as **fallback** so the demo always works.
* Keep **A1 outputs small** (top 10 sources, top 20–40 chunks) to keep latency low.
* Enforce **JSON-only** responses and always validate with Pydantic before using LLM output.
* Treat any citation not present in retrieval results as **invalid** → exclude it or mark **Unclear**.
* Save every prompt/response alongside run\_id for audit (beware PII).

If you want, I can turn this into a **task checklist** you can paste into your tracker, plus a short **README outline** for the repo and a 10-item QA script for the demo.