MinLaw 2 — Product & Engineering Specification (MVP)

A platform for fast, verifiable access to parliamentary statements, ministerial releases, government communications and media — with timeline views, cross-verification, and source-attributed answers.

---

0 — Executive summary & problem statement

Public discourse and legal/policy research in Singapore (and elsewhere) suffer from slow, manual verification of statements across parliamentary Hansard, press releases, ministerial social posts, and news media. Misquotes, out-of-context fragments and rapid misinformation spread make it harder to hold debates, do legal research, and make policy decisions.

MinLaw 2 MVP solves this by:

- continuously ingesting target sources (Hansard, press releases, government sites, official social posts, major news)
- extracting atomic statements, speakers, dates & context
- providing a fast hybrid search (keyword + semantic) that returns exact quoted spans with direct links and provenance
- tracking policy evolution as timelines and surfacing likely contradictions with confidence scores
- integrating simple legal estimation logic (e.g., LAB eBantu rules) as decision-support (clearly labelled, not legal advice).

LAB eBantu logic for SMU hackat···

Primary users: policy analysts, lawyers, journalists, civil society researchers, parliamentary staff.

Key principles: accuracy first (avoid hallucination), transparency (always show source), traceability (anchor quotes), and near-real-time freshness.

---

1 — Objectives & success metrics

Core objectives (MVP):

- Ingest 5 source classes daily (Hansard, ministerial press releases, official social media, government policy pages, 1 news publisher).
- Return search answers with exact quoted spans and link to the original source.
- Provide an interactive timeline per topic and a side-by-side contradiction UI.
- Automated contradiction detection that flags high-confidence candidates for human review.

Success metrics (example SLOs):

- Ingestion freshness: ⩾90% of target sources updated in last 24h.
- Indexing latency: median < 5 minutes from ingest → indexed (for incremental sources).
- Query latency: p95 < 600ms for simple queries.
- Precision (top-3) on QA set: ⩾80%.
- Human review false positive rate for contradiction flags: <25% initially, improving with feedback.

---

2 — High-level architecture (Mermaid)

```
flowchart LR
  subgraph Ingest
    Sched[Scheduler / Cron] --> Scraper[Scraper Agents]
    Scraper --> FetchQ[Fetch Queue]
    FetchQ --> Downloader[Downloader]
    Downloader --> RawStore[(S3 / Object Storage)]
    RawStore --> Parser[Parser & Extractor]
    Parser --> Normalizer[Normalizer & Splitter]
    Normalizer --> MetaDB[(Postgres)]
    Normalizer --> ES[(Elasticsearch)]
    Normalizer --> VectorDB[(Vector DB)]
  end

  subgraph Core
    API[Query API (Retriever+Reader)] --> Attribution[Attribution Service]
    API --> Tracker[Policy Tracker]
    API --> Contrad[Contradiction Detector]
    Tracker --> TimeSeries[(TimescaleDB)]
  end
```

```
subgraph Frontend
  FE[Next.js (Vercel)] -->|REST / WS| API
end

subgraph Ops
  Orch[Orchestrator: Airflow/Temporal] --> Scraper
  Orch --> Parser
  Logging[Prometheus + Grafana] --> Ops
end

RawStore --> Logging
API --> Logging
Contrad --> Logging
```

---

3 — System components & responsibilities

1. **Scheduler / Orchestrator**
    - Temporal / Airflow / Kubernetes CronJobs to trigger scrapers, parsers and downstream agents.
2. **Scraper / Downloader**
    - Respect robots.txt, use ETag / If-Modified-Since. Download HTML, PDF, RSS, social API outputs.
3. **Raw Store**
    - Object storage for immutable raw blobs (S3 / compatible).
4. **Parser & Normalizer**
    - HTML / PDF parsing, Readability, OCR fallback, canonicalize dates/speakers, split into atomic statements with offsets.
5. **Enricher**
    - NER, utterance speaker assignment, relation extraction, topic classification, generate embeddings (sentence/paragraph).
6. **Indexing**

- ElasticSearch for exact/keyword search + metadata; Vector DB (Weaviate/Pinecone/FAISS) for semantic retrieval.

7. **Query API**
    - Hybrid retriever (ES + VectorDB), optional LLM reader for concise summaries; always include raw quote spans + links.
8. **Policy Tracker**
    - Aggregate statements into timelines; supports filters per topic/speaker/date; persistent ChangeEvent store.
9. **Contradiction Detector**
    - NLI/entailment model to flag contradictions or significant position changes; compute confidence score.
10. **Frontend**
    - Next.js app on Vercel for public/internal UI; supports search, timeline, compare, document viewer and admin.
11. **Admin / Observability**
    - Source config UI, ingestion logs, DLQ viewer, dashboards (Prometheus + Grafana), error alerts.
12. **Human Review / Feedback loop**
    - Manual triage UI for contradiction flags to tune models and reduce false positives.

---

4 — Data sources (MVP scope & configs)

Initial MVP set (daily scheduled):

- Hansard transcripts (parliamentary debates) — high priority
- Ministerial press release pages (official ministry sites)
- Official ministerial social accounts (X/Twitter, Facebook pages) — via official APIs where possible
- Government policy pages & PDFs (statements, circulars)
- One national news outlet (e.g., Straits Times or CNA) as sample external verification

Source config model

- seed_urls, selectors (CSS / XPath), rate_limit, sitemap, auth (API keys), fetch_frequency, parser_profile, last_fetched_signature

---

5 — Data model (core tables) — examples & DDL

Below are simplified Postgres table schemas for the MVP.

documents

```
CREATE TABLE documents (
  id UUID PRIMARY KEY,
  source TEXT,
  url TEXT,
  raw_blob_uri TEXT,
  doc_type TEXT,
  published_at TIMESTAMP,
  fetched_at TIMESTAMP,
  fetch_signature TEXT,
  language TEXT,
  created_at TIMESTAMP DEFAULT now()
);
```

statements

```
CREATE TABLE statements (
  id UUID PRIMARY KEY,
  document_id UUID REFERENCES documents(id),
  text TEXT,
  start_offset INT,
  end_offset INT,
  speaker TEXT,
  role TEXT,
  statement_ts TIMESTAMP,
  embedding_id TEXT,
  created_at TIMESTAMP DEFAULT now()
);
```

topics

```
CREATE TABLE topics (
  id UUID PRIMARY KEY,
  canonical_name TEXT,
  aliases TEXT[],
  description TEXT
);
```

evaluations

```
CREATE TABLE evaluations (
  id UUID PRIMARY KEY,
  statement_id UUID REFERENCES statements(id),
  confidence FLOAT,
  contradiction_flags JSONB,
  reliability_score FLOAT,
  source_reliability FLOAT,
  derived_by TEXT,
  created_at TIMESTAMP DEFAULT now()
);
```

change_events

```
CREATE TABLE change_events (
  id UUID PRIMARY KEY,
  topic_id UUID REFERENCES topics(id),
  event_type TEXT,
  statement_ids UUID[],
  detected_at TIMESTAMP DEFAULT now(),
  metadata JSONB
);
```

(Keep additional admin tables: source_configs, ingestion_logs, users, manual_reviews.)

---

6 — Ingestion pipeline: step-by-step flow

1. **Scheduler triggers Scraper Agent** per source config.
2. **Scraper Agent** fetches page / API output:
    - compute fetch_signature = sha256(response_body); if same as previous → skip parse.
    - store raw blob in S3 with metadata.
    - push parse task to Parse Queue.
3. **Parser Worker**:
    - read raw blob, apply parser profile (HTML, PDF → text).
    - split to paragraphs and sentences; detect speaker lines for Hansard or press release headers.
    - annotate offsets and create statement records.
4. **Enricher**:
    - NER, topic classifier, compute embedding (embedding model), sentiment optional.
    - write metadata to Postgres, text to ES, embeddings to Vector DB.
5. **Post-processing**:
    - Contradiction Detector runs for new statements (or scheduled batch).
    - Policy Tracker updates timelines (ChangeEvents) if new position or contradiction detected.
6. **Notification**:
    - Emit events to frontend via websocket / pubsub for live update.

Operational features

- Idempotency tokens for tasks
- Dead-letter queue (DLQ) for parse failures with manual triage
- Rate-limiting and back-off to avoid IP bans
- Test harness for parser schema changes

---

7 — Retrieval & Query Engine (detailed)

Hybrid retrieval approach

- **Exact/Boolean Layer (ES):** for legal citations, phrase matches, date/classic lookups (high precision).

- **Semantic Layer (Vector DB):** for NL queries and contextual similarity (high recall).
- **Fusion & Ranker:** combine ES score + semantic similarity + freshness + source_reliability to produce final rank.

Reranker / Reader

- Optional small LLM (hosted / cloud) for short answering or rewriting, but always return:
    - exact quoted text (verbatim),
    - link to source + anchor (document id + char offsets),
    - provenance metadata (speaker, published_at, source reliability),
    - confidence score and explanation.

Confidence scoring (example)

confidence = normalize(w_agreement*agreement_score + w_reliability*source_reliability + w_recency*recency_score + w_sources*num_independent_sources)

Suggested weights (MVP):

- w_agreement = 0.4
- w_reliability = 0.3
- w_recency = 0.2
- w_sources = 0.1

(Weights configurable via admin.)

API — sample response (simplified)


```
{
  "query": "What did the Minister say about housing policy in 2024?",
  "top_answer": {
    "quote": "We will increase public housing subsidies starting July 2024...",
    "document_id": "uuid",
    "url": "https://...",
    "start_offset": 1243,
    "end_offset": 1299,
    "speaker": "Minister X",
    "published_at": "2024-07-01T09:00:00Z",
    "confidence": 0.82,
```

"supporting_snippets": [
{ "quote":"We will increase public housing...","url":"...","source_type":"hansard","score":0.8 },
{ "quote":"Press release: increased subsidies","url":"...","source_type":"press_release","score":0.65 }
]
}
}

---

8 — Contradiction detection & policy tracker

Algorithmic approach (MVP):

1. When a new statement S is ingested:
    - classify topic(s) using topic classifier.
    - retrieve candidate statements C on same topic (time window configurable).
2. For each candidate c in C:
    - compute NLI/entailment: model outputs {entailment, neutral, contradiction} with probability.
    - measure polarity change (e.g., "support" → "oppose") via simple rules + classifier.
3. If P(contradiction) > threshold or polarity flip, create a ChangeEvent linking S and c.
4. Compute confidence as weighted sum (see earlier).
5. Flag high-confidence events to human review UI before public/automated alerts.

Policy Tracker

- Groups ChangeEvents into an ordered timeline per Topic.
- Supports filters by speaker, ministry, source type, date range.
- Stores historical snapshots so users can see how wording changed over time.

UI experience

- Timeline with event markers; click to compare statements side-by-side with highlighted conflicting fragments and model explanation (e.g., "Different modal verbs: 'will' vs 'may' — deemed contradiction, score 0.84").

Human-in-loop

- Every high-impact contradiction (e.g., involving ministers) gets a manual verification step before being surfaced on alerts (configurable).

---

9 — LAB eBantu logic integration (Legal Aid Bureau)

We integrate the LAB regression rules as decision-support modules (clearly labelled "estimate / not legal advice") and log the calculation provenance for audit.

From LAB eBantu PDF (used for MVP calculation logic).

LAB eBantu logic for SMU hackat···

Example formulas (implement exactly as per document):

- Iddah_month = 0.14 * salary + 47 (rounding rules configurable; UI shows full precision and a rounded value)
    - lower = max(0, 0.14*salary - 3)
    - upper = 0.14*salary + 197
- Mutaah_day = 0.00096 * salary + 0.85
    - lower = 0.00096*salary - 0.15
    - upper = 0.00096*salary + 1.85

UI example: user enters salary S=3000:

- Iddah_month = 0.14*3000 + 47 = 467 (lower 417, upper 617).
- Mutaah_day = 0.00096*3000 + 0.85 = 3.73 (lower 2.73, upper 4.73). Show provenance: link to LAB PDF, show formula and rounding used.

LAB eBantu logic for SMU hackat···

Implementation note: Keep legal calculations auditable — store input, formula version, executor, timestamp and show "last updated" on UI.

---

10 — Frontend UX & screens (MVP)

1. **Search Home**
    - Natural language search box with examples.
    - Recent trending topics / tracked topics.
2. **Search Results**
    - Top answer with exact quote (highlighted) + "View source" anchor to document viewer.
    - Supporting snippets list sorted by confidence. Filters by date, speaker, source type.
    - "Track topic" CTA to follow updates/timeline.

3. **Document Viewer**
    - Full document with highlighted result snippet, jump-to-offset and copy permalink.
4. **Timeline (Policy Tracker)**
    - Interactive, zoomable timeline with markers for statements and ChangeEvents.
    - Click marker → side-by-side compare.
5. **Compare UI / Contradiction view**
    - Two (or more) statements side-by-side, highlighted differences, model explanation and confidence.
6. **Admin / Source Config**
    - Add/edit source, view ingestion logs, reprocess raw blob, inspect DLQ.
7. **LAB Calculator**
    - Simple UI with salary input and explained output (with provenance link).

Accessibility & UX principles

- Clear provenance shown for every factual claim.
- Avoid model-only claims: always pair LLM output with verbatim quotes.
- Provide “report error” button on every snippet for human feedback.

---

11 — API design (key endpoints)

- GET /api/search?q={q}&filters=… → ranked answers + snippets (see sample JSON above)
- GET /api/doc/{id} → document metadata + content + anchorable offsets
- GET /api/topic/{id}/timeline?from=&to= → timeline events and ChangeEvents
- GET /api/statement/{id}/contradictions → returns contradictions & evidence
- POST /api/admin/sources → add source config (admin)
- POST /api/admin/scrape/{source_id} → trigger scrape (admin)
- GET /api/health → service + dependencies health

Authentication

- OAuth2 for internal/admin; optional token for public read endpoints with rate limiting.

---

12 — Agentic workflows (detailed, idempotent & observable)

All agents must be idempotent, use task tokens, and report structured metrics.

Scraper Agent

- Trigger: scheduler or manual.
- Steps:
    1. Fetch seed URL (respect rate-limits).
    2. Compute sha256 of body; if signature unchanged → update fetched_at and exit.
    3. Save raw to S3; emit parse task (payload: raw_blob_uri, source_id, signature).
    4. Metrics: fetch_time, size_bytes, status.
- Error handling: 3 retries exponential backoff → mark source degraded & alert.

Parser Agent

- Trigger: parse queue.
- Steps:
    1. Load raw blob; choose parser profile (html/pdf/rss).
    2. Extract text, headers, speaker lines; split into statements with offsets.
    3. Create document + statements entries; on partial parse issues add to DLQ.
- Error handling: if repeated fails → file for manual review.

Enrichment Agent

- NER, topic classification, embedding generation.
- Writes embeddings to VectorDB, text to ES, metadata to Postgres.

Contradiction Detector Agent

- Trigger: new statements or scheduled batch.
- Steps:
    1. For each new statement, fetch candidates by topic from recent window.
    2. Run NLI model & polarity rules.
    3. If contradiction score > threshold → create change_event & pen to human review.
    4. Metrics: #checked, #flagged, FP rate (tracked by human feedback).

Policy Tracker Agent

- Maintains aggregated timeline for topics; updates TimescaleDB for fast timeline queries.

Cross-agent best practices

- All tasks include idempotency token, origin metadata, retry policy and link to raw blob for reproducibility.
- Use distributed tracing for visibility (OpenTelemetry).

---

13 — ML & models

Embedding model

- Use a compact embedding model that's fast & cost-effective for sentence-level (e.g., open models or managed embeddings).
  NLI / entailment
- Fine-tune a NLI model on policy / parliamentary style text for higher precision on contradictions.
  Topic classifier
- Lightweight transformer fine-tuned on labeled topics (e.g., housing, immigration, finance).
  Reader / summarizer
- Optional LLM for short contextual summaries — always show exact quote and links to source to limit hallucinations.

Training & feedback

- Use human-labeled dataset (from manual reviews) to fine-tune NLI and topic classifiers.
- Maintain a "gold" QA dataset of example queries + expected top-3 sources for regression testing.

---

14 — Privacy, legal & compliance

- **Copyright & takedown**: store only derivative text + direct link; implement takedown workflow for publishers (store immutable raw for audit).
- **Social APIs**: obey terms of use; store only permitted fields.
- **PII**: run PII detection and redact before public display (log redactions).
- **Legal disclaimers**: for LAB calculators and legal-adjacent features show "Not legal advice" and record calculations for audit.
- **Audit logs**: immutable logs for ingestion, parsing, contradictions and manual review actions.

---

15 — Observability, testing & QA

Monitoring

- Ingestion success/failure per source, indexing latency, p95 query latency, contradiction flag rate.
- Dashboards in Grafana; alerts for degraded sources or high parser error rates.

Testing

- Unit tests for parsers and enrichment.
- Integration tests: fetch→parse→index→search pipeline using sample raw files (Hansard transcripts).
- Regression: maintain labelled test queries & expected results.
- Human QA: review queue for contradictions; feed corrections into retraining pipeline.

---

16 — Security

- Secrets in Vault (AWS Secrets Manager / HashiCorp Vault).
- RBAC for admin actions.
- HTTPS everywhere; rate limiting; input sanitization.
- Use least-privilege IAM for S3/DB access.

---

17 — Deployment & CI/CD

Repo structure (monorepo suggested)

- /apps/frontend (Next.js)
- /services/query-api (Go or Node)
- /workers/scraper, /workers/parser (Python)
- /infra (Terraform / Helm charts)

Pipeline

- CI: unit tests, lint, container build.
- CD: push to registry → deploy to k8s (or Cloud Run); frontend to Vercel.
- Use staging / prod environments with separate data.

---

18 — MVP roadmap & prioritized milestones (deliverables, no time estimates)

Milestone 0 — Project setup

- Repo scaffold, infra skeleton (staging), object storage & DB provisioning.

Milestone 1 — Core ingest & storage

- Scraper for Hansard + ministerial press releases.
- Raw storage & parser that produces documents + statements.
- Admin UI to view ingestion logs & reprocess raw.

Milestone 2 — Search & attribution

- ES + VectorDB indexing.
- Query API returning exact quotes + link anchors.
- Frontend search results page.

Milestone 3 — Timeline & tracking

- Policy Tracker, ChangeEvent model, timeline UI.

Milestone 4 — Contradiction detection

- NLI service + human review workflow.
- Contradiction UI and alerting for high-confidence events.

Milestone 5 — LAB calculator + legal features

- Implement LAB formulas with audit trail UI.

LAB eBantu logic for SMU hackat···

Milestone 6 — Ops & extend sources

- Monitoring, scale, add social & news sources.

---

19 — Hackathon (SMU LIT) alignment & pitch framing

Use the SMU LIT Hackathon grading rubric to structure your presentation and demo. Key sections to highlight: problem, solution, technology, transformation potential, and overall impression. Show how each rubric item is addressed:

- **Problem (5)**: explain need for fast, verified parliamentary verification with real examples and friction points during debate or legal research.
- **Solution (5)**: demo search → exact quote → timeline & contradiction detection; show LAB calculator integration for Legal Aid Bureau.
- **Technology (5)**: present architecture diagram, parser demo, live query, and NLI contradiction detection.
- **Transformation Potential (5)**: explain how faster verification reduces misinformation, improves legal research efficiency and strengthens public discourse.

- **Overall Impression (5)**: polish the UI, narrate a user story (journalist verifies a quote live), show metrics and roadmap.

SMU LIT Hackathon Grading Rubri···

(Reference: grading rubric as uploaded).

SMU LIT Hackathon Grading Rubri···

---

20 — Acceptance criteria & example QA tests

Search correctness

- Given query about "housing policy 2024", top-3 contains Hansard/press release that includes the phrase; clicking shows quote anchored in document.

Timeline correctness

- When two conflicting minister statements exist, timeline shows both and a ChangeEvent with contradiction flag.

LAB calculator

- Given salary input, the UI shows formula, exact numeric output, lower/upper range, and link to LAB logic.

LAB eBantu logic for SMU hackat···

Contradiction triage

- Human-reviewed contradictions must reduce FP rate; record decisions and retrain model periodically.

---

21 — Risks & mitigations

- **Scraper fragility**: mitigated by schema profiles, XML watchers, admin reprocess and alerting.
- **Model hallucination**: show verbatim quotes + provenance; keep reader optional and conservative.
- **Legal copyright issues**: maintain link-back policy and takedown workflows.
- **False contradictions**: conservative thresholds + human review before surfacing to users.

---

22 — Next immediate actions (what I recommend you do right now)

1. Lock MVP source list for ingest (provide seed URLs and any available API keys).

2. Provide **2–4 sample raw files** (one Hansard transcript, one ministerial press release, one social post export, one example news article) to validate parsers.
3. Choose hosting preferences for Vector DB and embedding model (managed vs self-hosted).
4. Decide on an orchestrator (Temporal or Airflow recommended).
5. If you want, I'll generate: (a) Postgres DDL + ERD, (b) concrete API request/response JSON schemas for all endpoints, or (c) a starter repo skeleton (Docker + simple Hansard scraper + Next.js stub). Tell me which and I'll produce it next.

Appendix A — Quick reference: LAB formula examples (from uploaded PDF)

(See full logic and rounding rules in the LAB eBantu document). Example computed for salary=3000:

- Iddah_month = 0.14 × 3000 + 47 = 467 (lower 417, upper 617).

LAB eBantu logic for SMU hackat···

- Mutaah_day = 0.00096 × 3000 + 0.85 = 3.73 (lower 2.73, upper 4.73).

LAB eBantu logic for SMU hackat···