Product Requirements Document (PRD)

Brand Protection Monitor (PoC) — Source of Truth

Date: 2026-02-01 | **Version:** 1.0 (final)

Purpose: Single, authoritative functional specification + system logic + acceptance criteria to implement a Certificate Transparency (CT) monitoring PoC focused on early brand abuse detection.

Scale note: This agent’s standard typically produces 60+ pages. For this exercise, a shorter version is delivered, but it is significantly more detailed than the normalized context, designed so that another agent can consume only this document and execute downstream tasks without ambiguity.

Embedded context sources:

• Context Pack v1 (Normalizer): scope, modules, flows, rules, states, risks, and open questions.

• PRD structure template: sections 1–8 (Executive Summary → Screens/Flows).

Table of Contents

1. Executive Summary

One-liner: Full-stack web application for security/engineering teams that monitors a public CT Log and detects potential phishing domains or brand abuse via configurable keyword matching against certificate Common Name (CN) and Subject Alternative Names (SAN).

Elevator pitch: Instead of reviewing CT logs manually, the operator configures brand keywords and the system executes periodic cycles (fixed batch), highlighting actionable matches, exportable to CSV, with clear visibility into monitor state.

Segments and value propositions

Segment	Jobs-to-be-done	Value proposition	Measurable outcome
S1 — Security / AppSec operator	Detect suspicious TLS-issued domains early (before phishing campaigns).	Automated CT monitoring with direct CN/SAN + keyword matching; UI triage; CSV export.	Reduces time-to-detection (TTD) and helps prioritize investigation.
S2 — Platform engineering / SRE	Maintain reliable, operable signals without overwhelming systems or users.	Fixed batch + deduplication + monitor health + per-cycle metrics.	Less noise and higher pipeline observability.
S3 — Leadership (CISO / SecOps Lead)	Have evidence and reporting for decisions and internal audits.	Persisted (matches only), exportable, traceable results with key fields.	Repeatable, comparable reporting by period/cycle.

Business objectives

• Detect potential brand abuse (phishing / suspicious domains) from certificates observed in CT.

• Provide a web interface to manage keywords and review matches.

• Demonstrate technical feasibility of the ingest → process → highlight → export loop with Go/React/PostgreSQL.

Scope (MVP / PoC)

• Monitor a specific CT Log: https://oak.ct.letsencrypt.org/2026h2.

• Go backend retrieves a fixed, recent batch (recommended default: 100 entries) periodically (recommended default: every 60s).

• X.509 parsing to extract CN and SAN; substring matching against persisted keywords.

• Persist in PostgreSQL: keywords, matches, monitor state, and per-cycle metrics.

• React UI: keyword CRUD; state + metrics dashboard; match list with highlighting; CSV export.

Explicit non-goals

• Process the full CT log volume in real time (out of PoC scope).

• Store all CT certificates (only matching ones are stored).

• Automate response/remediation (takedowns, blocking, etc.).

1.1 Success Metrics (KPIs)

KPIs are defined as operable metrics per monitoring cycle and per period. While the PoC does not require targets, recommended targets are included for evaluation.

KPI-ID	Category	Name	Definition	Formula	Owner	Frequency	Recommended target (PoC)
KPI-RCH-01	Reach	Certificates processed (per cycle)	Entries evaluated in the last cycle.	count(entries_in_cycle)	Backend	Each cycle	≥ 100 (configurable)
KPI-RCH-02	Reach	Cycle latency	Time from cycle start until results are persisted.	t_persist - t_start	Backend	Each cycle	≤ 10s (batch 100)
KPI-ENG-01	Engagement	Active keywords	Number of keywords configured for monitoring.	count(keywords where status=active)	Operator	On-demand	5–50 in PoC
KPI-ENG-02	Engagement	Export usage	Number of CSV exports executed per day/week (utility signal).	count(exports)	Operator	Daily/Weekly	≥ 1/week in demo
KPI-TRX-01	Transactional	Certificates with match (per cycle)	Certificates whose CN or SAN contains at least one keyword.	count(matches_in_cycle)	Backend	Each cycle	N/A (depends on CT log)
KPI-TRX-02	Transactional	Operational precision (proxy)	Percent of matches marked as 'relevant' by the operator (if triage is added).	relevant/total	Operator	Weekly	≥ 30% on real dataset
KPI-OPS-01	Ops	Monitor state	Health indicator (idle/running/error) and time in error state.	state + time_in_error	Backend	Real-time	< 5 min in error without intervention

2. Problem Statement

Attackers register domains similar to brands and obtain TLS certificates, making phishing appear more legitimate. CT logs provide early visibility, but manual inspection does not scale or remain operable.

ID	Problem	Impact	Root cause	How the product solves it
P1	Late detection of suspicious domains issued with TLS certificates.	More successful phishing; brand reputation damage; losses.	Lack of early, automated monitoring.	CT polling + matching + persistence + visual triage.
P2	Manual CT log inspection is not operable for an end user.	High operational cost; missed signals.	Raw tooling; lack of UI and filters.	Dashboard + match list + CSV export.
P3	No visibility into processing state (health, volume, errors).	Unclear if the system works; wrong decisions.	No persisted metrics and states.	Monitor state machine + per-cycle metrics + status UI.

Product principles

- **Operability first:** the PoC must be usable by an operator without CT expertise.
- **Explainable signals:** each match must show why it matched (keyword + CN/SAN field).
- **Minimize noise:** deduplication and design to prevent repeating results between cycles.
- **Operational transparency:** user sees state, metrics, and errors without reading logs.
- **Technical simplicity:** fixed batch; one CT log; no out-of-scope features.

3. Target Users

Roles and permissions are defined for a PoC without authentication. Even without login, the design separates responsibilities to enable future phases.

3.1 Roles

Role	Description	Allowed actions (MVP)	Sensitivity
Operator (UI)	User who configures signals and reviews matches.	Keyword CRUD; view status/metrics; list matches; export CSV.	Medium
Backend service (Go)	Scheduler-driven process that consumes CT, parses, matches, and persists.	CT calls; X.509 parsing; persistence; expose REST API; generate CSV.	High
DB (PostgreSQL)	Source of truth for keywords, matches, and state.	Store and query entities; constraints; indexes.	High

3.2 Personas (behavioral)

Persona	Goals	Frustrations	Technical proficiency	Key needs
Persona A — SecOps analyst	Fast triage; export evidence; detect campaigns early.	Noise/duplicates; lack of context; unfriendly tools.	Medium/High	Clear highlighting, dedupe, filters, timestamps.
Persona B — AppSec / Platform engineer	Verify pipeline health; tune batch/cycle; avoid silent failures.	Errors not visible; missing metrics; uncontrolled retries.	High	Status dashboard, metrics, summarized error logs.
Persona C — Security leader	Simple reporting; actionable signals; demonstrate value.	Unexplainable results; inconsistent metrics.	Medium	Consistent KPIs, export, run traceability.

3.3 Experience map (Discovery → Loyalty)

• **Onboarding:** user opens UI, understands what CT is and why it matters (short explanation), adds keywords, validates the monitor runs.

• **Daily operation:** reviews dashboard, detects new matches (dedupe), filters, exports.

• **Iteration:** adjusts keywords (adds variants), reviews match trends, shares CSV.

• **Trust:** monitor state and per-cycle metrics show stability; user trusts the signal.

4. User Stories and Key Use Cases

Stories follow the TTI standard: Functional, Admin, System/Automation, and Security. For the PoC we assume a single user with no auth; security stories are still specified for minimal hardening.

ID	Category	Actor	Story	Acceptance criteria (AC)	Priority
HU-KW-01	Functional	Operator	As an operator, I want to create a keyword so the system detects domains containing that brand.	AC1: Reject empty/whitespace-only. AC2: Normalize (trim). AC3: Disallow duplicates via case-insensitive compare. AC4: Persist to DB and show immediately in UI.	P0
HU-KW-02	Functional	Operator	As an operator, I want to delete a keyword to reduce noise or remove obsolete signals.	AC1: UI confirmation. AC2: Delete or mark inactive (soft delete) with audit (deleted_at). AC3: Must not break historical matches (they remain).	P0
HU-KW-03	Functional	Operator	As an operator, I want to list and search keywords to manage the monitoring set.	AC1: Paginated list (if >50). AC2: Substring search. AC3: Show created_at and status (active/inactive).	P1
HU-MON-01	System/Automation	Backend	As a system, I want to execute a periodic CT consumption cycle to evaluate recent entries.	AC1: Configurable scheduler (default 60s). AC2: Each cycle records run_id, start/end, processed_count, match_count. AC3: If CT fails: state=error and retry next cycle.	P0
HU-MON-02	System/Automation	Backend	As a system, I want to calculate the entry range using STH to retrieve a fixed recent batch.	AC1: GET get-sth. AC2: tree_size defines end. AC3: start=max(0, end-batch_size). AC4: Define inclusive end. AC5: Persist start/end in monitor_run.	P0
HU-MON-03	System/Automation	Backend	As a system, I want to deduplicate matches across cycles to avoid repeating results.	AC1: Define unique key per certificate+keyword. AC2: Upsert—if exists, do not duplicate and update last_seen_at. AC3: Dashboard marks 'new' if first_seen_at is within the last cycle.	P0
HU-MAT-01	Functional	Operator	As an operator, I want to see a list of matches to investigate potential abuse.	AC1: List includes domain, matched_keyword(s), matched_field (CN/SAN), issuer, not_before/not_after, first_seen_at, last_seen_at. AC2: Highlight the matching fragment. AC3: Server-side pagination.	P0

ID	Category	Actor	Story	Acceptance criteria (AC)	Priority
HU-MAT-02	Functional	Operator	As an operator, I want to filter and sort matches to prioritize triage.	AC1: Filters: keyword, date (first_seen_at), issuer substring, 'new in last cycle'. AC2: Sort: first_seen desc (default), last_seen desc, domain asc.	P1
HU-EXP-01	Functional	Operator	As an operator, I want to export all stored matches to CSV for reporting.	AC1: Export button. AC2: Endpoint returns CSV with headers. AC3: Respects active UI filters (Option A selected). AC4: Streaming to avoid high memory.	P0
HU-DASH-01	Functional	Operator	As an operator, I want to see the monitor status to know whether it is processing.	AC1: Shows state idle/running/error. AC2: If error, show code + summarized message + last_error_at. AC3: Show last_run_at and last-cycle metrics.	P0
HU-SEC-01	Security	System	As a system, I want to validate and sanitize inputs to prevent injections and corrupted data.	AC1: keyword length 1..64. AC2: Escape UI output (XSS). AC3: Parameterized queries (SQLi).	P0
HU-SEC-02	Security	System	As a system, I want to limit endpoint abuse to protect the PoC.	AC1: Basic rate limiting (per IP) for export and search endpoints. AC2: Timeouts on external CT calls.	P1

5. Project Modules / Epics

The system is decomposed by modules (epics). Each module includes: purpose, screens/APIs, logic, rules, data, and edge cases.

5.1 Module: CT Log Consumer

Purpose: Connect to the public CT Log and retrieve recent entries via CT endpoints (get-sth, get-entries).

External dependency: Let's Encrypt Oak 2026h2 CT Log.

Inputs: base_url, batch_size, poll_interval, timeouts. **Output:** raw entries (leaf_input + extra_data) for parsing.

CT contract

• GET /ct/v1/get-sth → returns Signed Tree Head (tree_size, timestamp, sha256_root_hash, tree_head_signature).

• GET /ct/v1/get-entries?start={start}&end={end} → returns entries array (leaf_input, extra_data).

Range logic (closed recommendation)

• end = tree_size - 1 (last available index).

• start = max(0, end - (batch_size - 1)).

• If tree_size == 0: there are no entries; processed_count=0; state transitions running → idle.

Failure handling

• CT timeout (connect/read): set monitor to error with last_error_code=CT_TIMEOUT and summarized last_error_message; do not persist matches; next run retries.

• HTTP status != 200: CT_HTTP_ERROR with code.

• Invalid JSON response: CT_BAD_JSON; record payload size and truncate error message.

• Backoff: in PoC, retry only on the next cycle (no internal loops).

5.2 Module: Certificate Parser & Matcher

Purpose: Parse X.509 certificate to extract CN and SAN, and evaluate keyword matches (substring).

Closed decisions (to remove ambiguity)

• **Case-insensitive:** keywords and CN/SAN are compared in lowercase (Unicode case-fold if available).

• **Matching:** simple substring contains; no regex in PoC to avoid DoS patterns.

• **Cardinality:** a certificate may match multiple keywords; store one row per (cert_fingerprint, keyword).

• **Match field:** record whether the match occurred in CN, SAN, or both; also store the exact matched value.

Domain extraction

• CN: take subject.CommonName when it is DNS. If CN empty, consider SAN only.

• SAN: extract DNSNames from SubjectAltName extension.

• Normalization: trim, lowercase, remove trailing dot if present.

• Validation: ignore non-DNS entries (e.g., IPAddress) for the PoC.

Fingerprint (dedupe key)

Compute SHA-256 of the certificate leaf DER (or parsed leaf_input DER) and use it as cert_fingerprint.

Edge cases

• Unparseable certificate: increment parse_error_count and continue (must not crash the cycle).

• Very large SAN: limit to N=100 names to avoid memory spikes; record truncation flag.

• Internationalization (IDN): if punycode is detected, keep both representations (ASCII + Unicode) for display; matching is performed on ASCII punycode for consistency.

5.3 Module: Keyword Management

Purpose: CRUD monitored keywords from the UI; source of truth is PostgreSQL.

Business rules

BR-ID	Rule	Motivation	Recommended implementation
BR-KW-01	Disallow empty keywords and keywords > 64 chars.	Prevent junk data and large payloads.	Validation in API + CHECK constraint.
BR-KW-02	Disallow case-insensitive duplicates.	Prevent repeated signals and confusion.	Unique index on lower(value).
BR-KW-03	Soft delete (inactive) by default.	Maintain change traceability.	status, deleted_at fields.
BR-KW-04	Audit basic changes.	Debugging and accountability.	created_at, updated_at, deleted_at; optional actor='system'.

Impacted screens: Settings → Keywords. APIs: GET/POST/DELETE /keywords.

5.4 Module: Monitoring Dashboard (Frontend)

Purpose: Provide configuration and operational visualization interface.

Subcomponents

• Header: monitor state (badge) + last_run_at + Export CTA.

• Metric cards: processed_count, match_count, parse_error_count, ct_latency_ms, db_latency_ms.

• Matches table: server-side pagination, filters, highlighting.

• Keywords panel: inline CRUD + counter.

UX states

• Loading: skeletons for cards and table.

• Empty (no keywords): 'Add keywords to start' message.

• Empty (no matches): 'No matches yet' + tips.

• Error: banner with last_error_message and optional 'Retry now' button.

5.5 Module: Export (CSV)

Purpose: Export stored matches for external consumption.

Closed decision — CSV format

Column	Type	Description	Example
domain	string	Domain/DNS name where the match was observed (CN or SAN).	login-acme-example.com
matched_keyword	string	Keyword that triggered the match.	acme
matched_field	enum	cn | san | both	san

Column	Type	Description	Example
issuer	string	Certificate issuer name (Issuer CN/O).	R3 / Let’s Encrypt
not_before	timestamp	Validity start.	2026-02-01T10:20:30Z
not_after	timestamp	Validity end.	2026-05-02T10:20:30Z
cert_fingerprint_sha256	string	SHA-256 fingerprint (hex).	9f2a…
first_seen_at	timestamp	First time observed by the monitor.	2026-02-01T10:21:00Z
last_seen_at	timestamp	Last time observed (updated if it reappears).	2026-02-01T11:21:00Z
source_ct_log	string	CT log identifier (base_url or alias).	oak.ct.letsencrypt.org/2026h2

Export rules

• Export respects active UI filters (keyword, dates, issuer) for operability.

• Ordering: first_seen_at desc, domain asc, matched_keyword asc.

• Generation: streaming (row-by-row) to avoid loading all records into memory.

• Charset: UTF-8; delimiter: comma; quoting: RFC 4180.

6. Non-Functional Requirements

The PoC must be robust and reliable. These NFRs guide technical decisions without inflating scope.

ID	Category	Requirement	Verifiable criterion
NFR-PERF-01	Performance	Process batch_size=100 in ≤10s in local/dev environment.	monitor_run benchmark shows duration_ms ≤ 10000 (p95).
NFR-PERF-02	Performance	Paginated list endpoints respond ≤500ms (p95) with 10k matches.	Test with correct indexes; EXPLAIN ANALYZE.
NFR-REL-01	Availability	Monitor recovers from transient CT failures without intervention.	After simulated CT outage, next successful cycle returns to running/idle.
NFR-SEC-01	Security	Input validation/sanitization; basic abuse protection.	Rate limiting + parameterized SQL + UI escaping + external timeouts.
NFR-MNT-01	Maintainability	Modular code by responsibilities (consumer/parser/matcher/persistence/api).	Package structure; unit tests per module.
NFR-OBS-01	Observability	Record per-cycle metrics and last error; expose them via API and UI.	monitor_run + monitor_state tables and /monitor/status endpoint.
NFR-SCL-01	Scalability (PoC)	Configurable batch and poll interval without redeploy (env vars).	Changes via environment variables or config file (optional reload).

7. Assumptions, Clarifications, and Risks

No 'TBD' is allowed. Where the original context had gaps, recommended decisions are provided to enable implementation.

7.1 Closed assumptions (Strategist's Recommendations)

A-ID	Decision	Rationale	Impact
A-01	Case-insensitive matching.	Reduces false negatives; more intuitive UX.	Fewer misses; slight potential increase in false positives.
A-02	Substring matching (contains) without regex.	Simplicity and safety (avoid regex DoS).	Basic signal; may require more careful keywords.
A-03	Deduplication by (cert_fingerprint, keyword, matched_domain).	Avoids noise between cycles and supports multi-domain SANs.	Cleaner dashboard; last_seen_at can be updated.
A-04	Monitor state persisted in DB (monitor_state) + historical runs in monitor_run.	Avoids blind reprocessing and enables consistent UI.	Resilience and traceability.
A-05	CSV export respects UI filters.	Operability (export only what the user is analyzing).	Smaller files; higher utility.

7.2 Risks and mitigations

Risk	Prob.	Impact	Signal/Detection	Mitigation	Plan B (PoC)
Variable CT log volume/rate.	Med	Med	duration_ms increases; timeouts.	Fixed batch; timeouts; per-cycle backoff.	Reduce batch_size or increase poll_interval.
X.509 parsing fails for some entries.	Med	Low/Med	parse_error_count increases.	Skip entry; summarized log; metric.	List failed fingerprints for debugging.
Duplicates between cycles (noise).	High	Med	Same domain/keyword repeats.	Upsert with unique constraint; update last_seen_at.	'New' flag only if first_seen_at in last cycle.
DB unavailable.	Low/Med	High	DB error; state=error.	Per-cycle retries; healthcheck; timeouts.	Degraded mode: do not persist; show error.
UI slow with many matches.	Med	Med	p95 > 500ms.	Server-side pagination; indexes; limit columns.	Async export (Phase 2).

8. Screens and User Flows

Detailed screens: layout, interactive components, state logic, and data binding. PoC assumes a single web app (no public portal).

8.1 Screen inventory

Screen ID	Name	User	Objective	APIs used
SCR-01	Dashboard	Operator	View state, metrics, and the matches list.	GET /monitor/status; GET /matches
SCR-02	Keywords	Operator	Manage keywords (CRUD).	GET/POST/DELETE /keywords
SCR-03	Export Modal	Operator	Configure export (filters) and download CSV.	GET /export.csv

8.2 Screen Spec: Dashboard (SCR-01)

Visual layout (responsive)

• Row 1: Title + state badge (idle/running/error) + last_run_at timestamp.

• Row 2: 4 metric cards: processed_count, match_count, parse_error_count, cycle_duration_ms.

• Row 3: Matches table with filter bar (keyword, issuer, date range, 'new').

Interactive elements

• Keyword filter (multi-select dropdown).

• Domain/issuer search (input).

• Toggle: 'Only new in last cycle'.

• Export button (opens SCR-03).

• Server-side pagination: page, page_size (10/25/50).

Data binding

• On load: fetch /monitor/status and /matches?page=1&page_size=25&sort=first_seen_desc.

• On filter change: debounce 300ms and refetch /matches with query params.

• On export: open modal with current filters pre-filled.

State logic

• Loading: skeleton for cards and table.

• Error: persistent banner if /monitor/status.state=error or /matches fails; shows message and retry button.

• Empty (no keywords): CTA to SCR-02.

• Empty (no matches): educational copy + keyword tips.

8.3 Screen Spec: Keywords (SCR-02)

Layout

• Input + 'Add' button at top.

• Keyword list with actions (delete) and timestamps.

• Inline validation indicator (duplicate, empty, too long).

Validations

• Trim and lower preview; store original value + normalized_value (lower) for indexes.

• API errors shown inline and must not clear user input.

Edge cases

• If a keyword used by historical matches is removed, matches remain; UI may show it as 'disabled keyword' if queried.

• If backend monitor is in error, keyword CRUD should still work (if DB is OK).

8.4 Screen Spec: Export Modal (SCR-03)

Behavior

• Modal does not navigate away from dashboard.

• Shows summary of active filters (chips).

• Download CSV button triggers /export.csv?{params}.

• During download: 'Preparing...' state and disable button to avoid duplicates.

9. System Architecture and Data Model

Layered architecture: React+TS UI ↔ Go REST API ↔ PostgreSQL. External integration: public CT Log. System runs locally or in a PoC environment.

9.1 Logical diagram (text)

Scheduler (Go) → CT Consumer → Parser/Matcher → Persistence (Postgres) → REST API → UI

9.2 Entities and tables (PostgreSQL)

The following DDL is proposed (engineering reference):

```
-- keywords
CREATE TABLE keywords (
  keyword_id BIGSERIAL PRIMARY KEY,
  value TEXT NOT NULL,
  normalized_value TEXT NOT NULL,
  status TEXT NOT NULL DEFAULT 'active', -- active|inactive
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  deleted_at TIMESTAMPTZ
);
CREATE UNIQUE INDEX ux_keywords_normalized ON keywords (normalized_value) WHERE deleted_at IS NULL;

-- monitor_state (single row)
CREATE TABLE monitor_state (
  id SMALLINT PRIMARY KEY DEFAULT 1,
  state TEXT NOT NULL DEFAULT 'idle', -- idle|running|error
  last_run_at TIMESTAMPTZ,
  last_success_at TIMESTAMPTZ,
  last_error_at TIMESTAMPTZ,
  last_error_code TEXT,
  last_error_message TEXT,
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

-- monitor_run (append-only)
CREATE TABLE monitor_run (
  run_id BIGSERIAL PRIMARY KEY,
  started_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  finished_at TIMESTAMPTZ,
  ct_tree_size BIGINT,
  range_start BIGINT,
  range_end BIGINT,
  processed_count INT NOT NULL DEFAULT 0,
  match_count INT NOT NULL DEFAULT 0,
  parse_error_count INT NOT NULL DEFAULT 0,
  duration_ms INT,
  ct_latency_ms INT,
  db_latency_ms INT,
  state TEXT NOT NULL DEFAULT 'running', -- running|success|error
  error_code TEXT,
  error_message TEXT
);

-- matched_certificates (one per (fingerprint, keyword, domain))
CREATE TABLE matched_certificates (
  match_id BIGSERIAL PRIMARY KEY,
  cert_fingerprint_sha256 TEXT NOT NULL,
  matched_domain TEXT NOT NULL,
  matched_keyword TEXT NOT NULL, -- store denormalized for export speed
  matched_field TEXT NOT NULL, -- cn|san|both
  issuer TEXT,
  not_before TIMESTAMPTZ,
  not_after TIMESTAMPTZ,
  source_ct_log TEXT NOT NULL,
  first_seen_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  last_seen_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  last_run_id BIGINT REFERENCES monitor_run(run_id)
);
CREATE UNIQUE INDEX ux_match_key ON matched_certificates (cert_fingerprint_sha256, matched_keyword,
 matched_domain);
CREATE INDEX ix_matches_first_seen ON matched_certificates (first_seen_at DESC);
CREATE INDEX ix_matches_keyword ON matched_certificates (matched_keyword);
CREATE INDEX ix_matches_domain ON matched_certificates (matched_domain);
```

9.3 Data design notes

- matched_keyword is denormalized in matched_certificates for faster export (avoids joins).
- monitor_state stores current state for UI; monitor_run stores history for analysis/debugging.
- Unique constraints implement deduplication; last_seen_at is updated via UPSERT.
- Indexes are aligned to typical queries: first_seen desc (default), filters by keyword and domain.

10. REST API Specification (Contracts)

The API is REST, JSON. Primary endpoints: keywords, monitor status, matches, export. Contracts include request/response shape, validations, and error codes.

10.1 GET /monitor/status

Field	Type	Description
state	string	idle | running | error
last_run_at	timestamp	Last run (start time).
last_success_at	timestamp	Last successful run.
last_error_code	string	Error code (if applicable).
last_error_message	string	Summarized error message (if applicable).
metrics_last_run	object	processed_count, match_count, parse_error_count, duration_ms, ct_latency_ms, db_latency_ms.

10.2 GET /keywords

Field	Type	Description
items	array	List of keywords (active/inactive).
total	int	Total for pagination (if applicable).

10.3 POST /keywords

Input	Type	Validation
value	string	1..64 chars; trim; no duplicates (case-insensitive).

10.4 DELETE /keywords/{keyword_id}

Behavior	Details	Notes
Soft delete	Sets deleted_at and status=inactive.	Does not delete historical matches.

10.5 GET /matches

Query param	Type	Description
page	int	≥ 1
page_size	int	10 | 25 | 50 (default 25)
keyword	string	Exact filter by matched_keyword (optional)
q	string	Substring search in domain or issuer (optional)
from	date	Filter first_seen_at ≥ from (optional)

Query param	Type	Description
to	date	Filter first_seen_at ≤ to (optional)
new_only	bool	Only first_seen_at within the last run (optional)
sort	string	first_seen_desc (default) | last_seen_desc | domain_asc

10.6 GET /export.csv

Query params: same as /matches — reuses filters for export.

10.7 Standard error codes

HTTP	error_code	When it occurs	Recommended UI action
400	VALIDATION_ERROR	Invalid input (keyword, params).	Show inline message; highlight field.
404	NOT_FOUND	keyword_id does not exist.	Toast: 'Not found'.
409	DUPLICATE	Duplicate keyword (case-insensitive).	Message: 'Already exists'.
429	RATE_LIMITED	Too many requests.	Retry with backoff; show notice.
500	INTERNAL_ERROR	Unexpected failure.	Error banner + suggest retry.
503	DB_UNAVAILABLE	DB down or timeout.	State=error; disable DB-dependent actions.

11. Business Rules, State Machines, and Automations

This section defines hard system logic: rules, states, and automated triggers.

11.1 Rules catalog (BR-xxx)

BR-ID	Rule	Applies to	Input	Output/Decision	Edge cases
BR-001	Keywords must be persisted in PostgreSQL.	Keyword Mgmt	value	Insert/Update	Duplicates via constraint.
BR-002	Matching evaluates CN and all SAN.	Matcher	CN, SANs, keywords	match/no match + field	SAN empty; CN empty.
BR-003	Backend retrieves a fixed recent batch periodically; does not process full log.	CT Consumer	STH + config	entries per cycle	Small tree_size.
BR-004	Only certificates with match are stored.	Persistence	matching result	UPSERT matched_certificates	Multi-match; dedupe.
BR-005	UI shows monitor state and last-cycle metrics.	Dashboard	monitor_state + last run	cards + badge	Partial errors.
BR-006	CSV export available for stored matches with filters.	Export	filters	CSV stream	Large dataset.
BR-007	Deduplication by unique key and last_seen_at updates.	Persistence	match key	upsert	Collisions unlikely.
BR-008	If CT fails or DB fails, monitor enters error and the cycle is not successful.	Monitor	exception	state=error	Parse errors are non-fatal.

11.2 State machine: Monitor

States: idle, running, error.

From	Event	To	Actions
idle	scheduler_tick	running	Create monitor_run(state=running); set monitor_state.state=running.
running	cycle_success	idle	Update monitor_run(state=success); update monitor_state.last_success_at; set idle.
running	cycle_fatal_error (CT/DB)	error	Update run(state=error, error_code/message); set monitor_state error fields.
error	scheduler_tick	running	Retry cycle; on success → idle.

Non-fatal errors: parse_error_count increments, but the cycle may still be successful.

11.3 Automations (system triggers)

AUTO-ID	Trigger	Condition	Action	Observability
AUTO-01	scheduler_tick	Each poll_interval	Start CT cycle (HU-MON-01).	monitor_run.started_at
AUTO-02	ct_fetch_complete	Valid get-entries response	Process and match entries.	processed_count, ct_latency_ms
AUTO-03	match_found	CN/SAN contains keyword	UPSERT match; update last_seen_at; set first_seen_at if new.	match_count; new_only derivable
AUTO-04	cycle_end	After processing	Persist metrics and duration_ms; update monitor_state.	monitor_state.last_run_at

12. Observability, Operations, and Rollout Plan

Defines how to operate the PoC, detect failures, and demonstrate value.

12.1 Logging and metrics

• Structured JSON logs in backend with correlation: run_id.

• Minimum fields: run_id, event, duration_ms, processed_count, match_count, error_code.

• Persist metrics in monitor_run for UI and debugging without depending on external logs.

12.2 Health checks

• GET /healthz: verifies process is alive.

• GET /readyz: verifies DB connectivity and ability to query monitor_state.

• UI: show banner if /readyz fails.

12.3 Rollout plan (PoC)

Phase	Objective	Exit criteria	Artifacts
F0 — Local dev	Full loop with real CT and local DB.	Successful cycle + UI lists matches.	docker-compose, seed keywords
F1 — Demo	Operable demo for stakeholders.	CSV export works with filters; dashboard stable for 30 min.	demo script, sample dataset
F2 — Hardening	Minimum robustness for internal use.	Rate limit + timeouts + verified dedupe.	tests, ops docs

12.4 Suggested roadmap (out of PoC scope)

• Phase 2: multiple CT logs; incremental state (not only recent batch).

• Proactive alerts (email/webhook) and risk scoring.

• Authentication (SSO) and multi-tenant.

• Triage: mark match as 'relevant'/'false positive' to refine rules.

13. PDF Format Validations

This document was generated using a flowables-based layout engine, guaranteeing:

• No content overlap: each block is positioned sequentially based on its real rendered height.

• Tables use Paragraph-based cells (wrapping) and split across pages with repeating headers.

• Margins (36pt) are respected on all pages; no content is drawn outside the printable area.

• Padding and TOP valign prevent text from being cut inside table cells.

• PageBreaks are used only on major section changes; otherwise content flows automatically.

Recommended visual verification checklist (for the consuming agent or human review)

• Verify table headers repeat when a table continues on a new page.

• Confirm no cell contains text outside its borders.

• Confirm there are no overlapped lines or text on top of other elements.

• Validate timestamps/long fields wrap in API/DDL tables.

Requested inputs for a future iteration (process requirement)

• **Core Business Vision:** which brand(s) and which risks are prioritized (phishing, typosquatting, brand impersonation).

• **User Ecosystem:** who consumes exports and what downstream actions they execute.

• **Monetization/Impact Model:** how impact is measured (TTD, incidents avoided) and who sponsors the product.