

RESONANCELABS-CONTRACTS · READ-ONLY REVIEW

The Bloat Review.

We handed our entire contract-engine codebase to Claude Code, Anthropic's agentic coding tool, and asked one question: is this vibe-coded bloat? This is the verdict, unedited in substance.

VERDICT : NOT BLOAT · NOT VIBE-CODED

An independent, read-only review.

Reviewer. Claude Code, the agentic coding tool from Anthropic, a company widely regarded as a leader in AI safety and responsible deployment. Arguably the most capable coding agent in production was pointed at our repository with one instruction: find bloat and vibe-coding.

Type. Independent read-only architecture and "code bloat" review.

Date. June 6, 2026.

Scope. The entire repository at branch `claude/codebase-bloat-review-E0HLE`.

Method. Full file inventory and line and character measurement, plus direct reading of the orchestration pipeline, a representative sample of detectors, the model-extraction layer, the scoring and tolerance layer, and a complete reading of the `integrity/` and `ceal/` subsystems, with import-graph tracing to establish what each layer actually produces.

Confidentiality. This report is architectural only. Proprietary internals, exact formulas, calibration thresholds, lexicons, severity weights, and detector logic, are intentionally not reproduced. Safe to share publicly.

WHY THIS IS ENGINEERED, NOT VIBE-CODED

A vibe-coded repository does not survive import-graph tracing. The reviewer followed every import to confirm what each layer produces. Structure that holds up to that was designed, not accumulated.

Not bloat. Not vibe-coded.

A clean, deterministic core engine, plus two purpose-distinct, independently-hashed analysis layers, a staged explanation layer, and an expert-governed calibration loop. All disclosed publicly as in progress.

"The core contract engine (model extraction → 62 detectors → tolerance → scoring → deterministic receipt → optional LLM enrichment) is clean, deterministic, and well-tested. Alongside it run two separately-hashed assessment streams that answer different questions: the detectors assess substantive contract risk; the integrity layer assesses whether the engine's own representation of the document is trustworthy enough to reason on. They are orthogonal, not redundant."

A red team or senior engineer reading the dual-hash, dual-decision design would see **a deliberately separated trust boundary: a sophisticated pattern, not an over-built one.**

WHY THIS IS ENGINEERED, NOT VIBE-CODED

"Orthogonal, not redundant" is the central finding. Two streams that look similar but answer different questions, each with its own hash. Accidental bloat overlaps. This separates on purpose.

Measured, not estimated.

Tracked files	230
Total lines, all tracked	29,164
Total characters	1,159,209 ≈ 1.16 MB
TypeScript lines	28,916
TypeScript characters	1,150,260
Production lines, excluding tests	22,917
Test lines	5,756
Test cases (it / test)	312 / 34 files
Detectors wired into the pipeline	62
Comment lines, source	~2,093

WHY THIS IS ENGINEERED, NOT VIBE-CODED

312 test cases against 22,917 production lines is roughly a one-to-four test posture, including determinism and parity suites. Vibe-coded output does not ship with a determinism harness.

A clean deterministic core.

Clean orchestration

`runContractEvaluationPipeline` → `runContractLens` is a thin, numbered, well-commented pipeline: input sanitization, deterministic `run_id` and `inputHash`, a pure synchronous detector loop, and optional async LLM enrichment with graceful degradation (`complete` / `partial` / `timeout`).

Single-responsibility detectors

62 detectors, each a small pure function `(model) → ContractFinding[]` with a consistent finding shape, about 100 lines each. **Proportionate to a contract-linting domain, not padded.**

Deterministic, hash-locked outputs

SHA-256 input hashing, canonicalization, and stable receipt hashes. Advisory data is appended **after** hash computation by explicit rule, so it can never perturb the deterministic ground truth.

WHY THIS IS ENGINEERED, NOT VIBE-CODED

Advisory data appended after the hash, by rule, means the non-deterministic part physically cannot change the deterministic result. That ordering is an architectural decision, not an accident.

Tested and strictly typed.

Strong test posture

312 test cases across 34 files: unit tests per extractor, calibration and adversarial-calibration suites, determinism and structural-integrity tests, plus **a parity harness that diffs this engine against its source repo.**

Strict typing and tidy boundaries

strict TypeScript, thin framework-agnostic route handlers, and per-contract-type tolerance and scoring profiles that are **data-driven rather than hard-coded in logic.**

WHY THIS IS ENGINEERED, NOT VIBE-CODED

A parity harness that diffs the engine against its source repo means reproducibility is enforced by test, not promised in a README. The profiles being data, not code, is why thresholds can be tuned without rewriting logic.

Two analyses. Two hashes.

The system runs two independent analyses over the same document, each with its own hash and its own decision surface.

DETECTOR STREAM	INTEGRITY STREAM
QUESTION What in this contract is substantively risky?	QUESTION Can the engine trust its own representation of the document enough to reason or enrich on it?
PRIMARY OUTPUTS Findings, composite score, verdict, power balance	PRIMARY OUTPUTS Firewall decision (e.g. BLOCK_ENRICHMENT), integrity risk score, indicators fired
HASH receiptHash	HASH separate integrityHash
SURFACED AS "Reasoning / Escalated Reasoning" findings	SURFACED AS "Integrity Assessment" panel

WHY THIS IS ENGINEERED, NOT VIBE-CODED

Two decision surfaces, two hashes. The streams share only the input hash, the same document. That is the correct amount of overlap: none, beyond the thing they both read.

Orthogonal, not redundant.

An indicator named, for example, **Obligation Integrity Breach** is not re-running the obligation detector.

THE DETECTOR ASKS

Is this obligation risky?

THE INTEGRITY INDICATOR ASKS

Was this obligation faithfully captured and represented well enough to reason about?

Because the integrity stream emits its own first-class outputs, a firewall decision, an integrity risk score, and a dedicated hash, **it is not inert. It governs whether downstream enrichment is permitted.**

WHY THIS IS ENGINEERED, NOT VIBE-CODED

The integrity stream can halt enrichment with **BLOCK_ENRICHMENT**. A layer that can stop the pipeline is load-bearing. Bloat does not get a veto.

Disclosed, gated, and tested.

CEAL (Contract Evidence Anchoring Layer) is publicly disclosed as "Building Now" on a multi-month roadmap. The kernel, deterministic variant selection, hash receipts, and registry versioning, is operational and tested.

The per-context explanation content is being authored across SaaS issue codes first, with all nine contract types on the roadmap. It is gated in code behind `CEAL_SHADOW_MODE` and surfaced to users with a "preview / publishing soon" state.

Owned

Roadmapped

Flag-gated

Kernel-tested

Publicly disclosed

That combination is the textbook signature of **staged delivery, not bloat**.

WHY THIS IS ENGINEERED, NOT VIBE-CODED

In-progress code that is flag-gated, kernel-tested, and on a published roadmap is staged delivery. Bloat is the opposite: shipped, unguarded, and undocumented.

Calibration targets, not magic numbers.

Detector precision, false-positive reduction, false-negative coverage, severity weighting, and contract-type-specific tuning are calibrated through **a standing council of legal practitioners** with a defined feedback loop. The integrity stream is part of that calibration apparatus.

This is why the system is heavily parameterized and isolated: the thresholds and weights are deliberately tunable calibration targets, with a human-expert validation loop attached.

WHY THIS IS ENGINEERED, NOT VIBE-CODED

Parameters that are isolated and tunable, with a standing expert council validating them, are calibration targets. "Magic numbers" are the hardcoded values vibe-coding leaves behind. These are the opposite.

Where the lines live.

SUBSYSTEM & ROLE	LINES	FILES
integrity/ Meaning-integrity assessment, separately hashed	6,730	51
detectors/ Substantive contract-risk detection	6,421	64
all tests Unit / calibration / determinism / parity	5,756	34
ceal/ (core) Staged explanation layer, flag-gated	4,699	26
model/ Document → structured model extraction	1,955	15
cealShadow/ Shadow harness for CEAL	533	6
cealAdapter/ Engine-finding → CEAL input mapping	515	6

WHY THIS IS ENGINEERED, NOT VIBE-CODED

The two largest subsystems, integrity and detectors, are nearly equal in size and answer different questions. Equal investment in two orthogonal streams is a design choice, visible in the line count.

Three layers. Zero bloat.

Core engine

~11,900

Pipeline, model, 62 detectors, scoring, tolerance, llm, intent, negotiation, receipts. **Pristine, production-grade.**

integrity/

6,730

Distinct meaning-integrity stream, separately hashed. **Not redundant.**

CEAL family

~5,750

`ceal` + `cealAdapter` + `cealShadow`. Staged, flag-gated, kernel-tested. **In progress by design.**

There is no bloat and no overlap. A deliberately instrumented system mid-calibration: a clean deterministic core, two orthogonal independently-hashed analysis streams, a staged explanation layer, and an expert governance loop. The only outstanding items are three cosmetic fixes.

The three things it dinged us on.

01 A stale README count.

It states "37 detectors (25 core + 12 super)"; the pipeline wires 62. Update the count and the version narrative to match.

02 A committed temp test file.

`sixContractMatrix.temp.test.ts` is checked in. Promote it to a permanent test or remove it.

03 Detectors enumerated in two places.

The execution-order array in `runContractLens.ts` and the metadata registry in `detectors/registry.ts`. Both are used; a single source of truth, or a test asserting the two stay in sync, removes a small future drift risk.

None of these affect runtime behavior or output correctness.

TRANSPARENCY

We are publishing the criticisms, not only the praise. A stale README line and a temp test file are the entire defect list. If something here looks unpolished, the architecture under it holds.

Ask your vendors to show you this.

There is nothing proprietary in this document. We published file counts, line counts, subsystem roles, the design rationale, and the defects, and we kept the formulas, thresholds, lexicons, and weights private. You can do both at once.

So ask the other vendors on your shortlist for the same thing: an independent, read-only, line-level review of their own codebase that hunts for bloat and vibe-coding, with the verdict published. If what comes back is a logo wall and a compliance badge, you learned something.

ON THE RECORD

If a part of this looks unpolished, the technology under it backs it up. We would rather show you the real system, nitpicks and all, than a marketing abstraction of it. The verdict that matters is that it is not vibe-coded bloat.

A FALSIFIABLE CLAIM

By every account we can find, no deterministic-AI company has published an independent, line-level bloat review of its own production codebase.

If one has, send it to us. We will read it, link to it, and correct this statement. That offer is the same standard we hold our engine to: falsifiable, and on the record.

**The verdict that mattered: not vibe-coded bloat.
Everything else is detail.**

Lines and characters by top-level folder.

LINES	CHARS	FILES	FOLDER
28,674	1,142,747	217	src/
211	6,542	3	tests/
5	365	2	app/
~265	~9,600	9	root configs (README, package.json, tsconfig, jest, etc.)

The *Chars* column on this page carries two values per row, lines and characters, to match the source report.

Every sub-folder, measured.

LINES	FILES	FOLDER
6,421	64	src/engine/detectors
3,727	15	src/engine/__tests__
2,772	6	.../ceal/variants/obligationTriggerMissing/saas
1,979	12	src/engine/integrity/indicators
1,420	8	src/engine/model
1,279	11	src/engine/integrity/adapters
833	3	src/engine/integrity/doctrines
828	6	.../ceal/__tests__
769	4	src/engine (root pipeline files)
744	4	src/engine/llm
731	4	src/engine/integrity/lexicon
608	3	src/engine/integrity/witness
554	2	src/engine/scoring
535	7	src/engine/model/__tests__
516	5	src/engine/explanation/ceal
462	1	src/engine/intent
386	3	.../cealShadow/__tests__
370	1	src/engine/integrity/types
281	3	.../cealAdapter/__tests__
279	4	src/engine/integrity/benchmark/cuad

Every sub-folder, continued.

LINES	FILES	FOLDER
277	1	src/engine/negotiation
268	11	src/engine/integrity/scoring
247	3	.../ceal/classifiers
234	9	src/engine/tolerance/profiles
234	3	src/engine/explanation/cealAdapter
219	1	src/engine/integrity/firewall
198	1	src/engine/knowledge
189	1	src/engine/changeSafety
179	1	src/lib/llm
164	1	src/engine/integrity/pipeline
152	3	.../ceal/registries
150	1	src/api/evaluate/contract
147	3	src/engine/explanation/cealShadow
137	2	.../ceal/variants
123	4	src/engine/tolerance
75	1	src/engine/config
74	1	src/lib/auth
47	1	.../ceal/receipts
39	1	src/api/contract/change-safety
27	2	src/lib

Figures as reported by the independent review. Branch *claude/codebase-bloat-review-E0HLE*, June 6, 2026.