CASE STUDY · ELEMZA · SEPTEMBER 2026

Rebuilding a regulated platform in 70 days, *and the system that kept the AI agents honest*

eIEMZA verifies identities and carries legally binding signatures for Iranian businesses. This is how it was rebuilt, what it has done since, what broke on the way, and the engineering method that made building it with AI coding agents safe enough to sign a contract with.

Arash Banaeian
Systems Architect · AI and Data Systems · Fractional CTO
Sole technical owner of eIEMZA since January 2026
arash.bne@gmail.com
elemza.com/Arash

4.7×
contracts in 25 days of September against the old platform's best full month
1,747 against 372

81.9%
of contracts since relaunch completed by every signer
median 7.5 minutes from creation to last signature

71
contracts carried through identity-registry outages by the retry queue
60 of them went on to completion

99.98%
of 1.04M requests served without a server error
25 Aug to 25 Sep 2026

THE ARGUMENT

In January 2026 the old platform had all but stopped: four contracts that month. I was brought in to rebuild it, and I did not repair it. I specified it again from its business and regulatory contract, never opening the old code, and built it in a new codebase. It went live on 21 April, 70 days after the first commit, carried two years of live data across in a two-minute cut-over, and on 5 September passed the old platform's lifetime total of 4,068 contracts.

AI coding agents wrote the code. My work was the system around them: rules every agent loads, hooks that enforce them, a law that no claim counts without its evidence, a separate adversarial review of every substantial change, a deploy pipeline that refuses, and a catalogue of 100 real failures, each turned into a mechanical guard. The pages that follow show that system, the numbers it produced, and the places where it still falls short.

READ IT IN TEN MINUTES

01 A stalled product, rebuilt from its contract rather than its code

CONSTRAINTS **What made it hard**

- **A signature is a legal act** and a lost row is a lost contract, so every change touches something a court may read.
- **Identity must be checked against the national registry** before anyone may sign, and that registry drops several times a day.
- **Sanctions block** part of the software supply chain; the production server was patched in place for months.
- **One technical owner**, and a workforce of AI agents that are fast, tireless and confidently wrong in ways that look exactly like being right.

DECISION **Why rebuild, not repair**

Repairing inherits every hidden assumption of the old code. Rebuilding from the contract means writing down what the business, the law and the data actually require, then building exactly that. It cost nothing in continuity: the data came across with every identifier intact, so every old link, signer and invoice still resolves.

It also made the result checkable. Each requirement became a rule, a test or a guard that a reviewer can open, rather than a behaviour hidden in someone's memory.

From a one-page brief to the old platform's lifetime total

TIMELINE

Jan 2026 — The old platform is down to four contracts a month. I am brought in to rebuild it.

10 Feb — First commit in a new codebase. The old code is never used.

18 Apr — 18,470 rows in nine tables moved with identifiers intact, in a two-minute cut-over.

21 Apr — First contract on the new platform: day 70.

11 Jun — Qualified PKI signatures live through a licensed certificate authority.

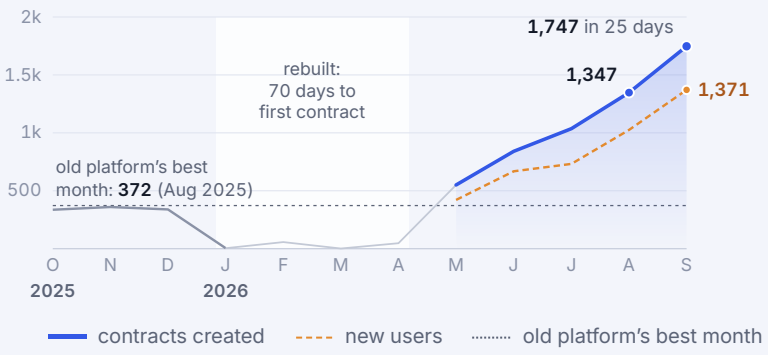
5 Sep — Passes the old platform's lifetime total: 4,068 contracts over 23 months.

6 Sep — First signer pays their own share under the new counterparty-pays model.

25 Sep — 1,747 contracts so far this month.

The result, before any marketing

MONTHLY, MEASURED IN PRODUCTION



1–25 SEPTEMBER	OLD BEST	2026	×
Contracts	372	1,747	4.7×
New users	298	1,371	4.6×
Paying customers	16	97	6.1×
Money collected			3.8×

Each old figure is the old platform's best single month for that measure, over its whole life (May 2024 to April 2026). September is 25 days against a full month. Revenue is shown only as a multiple. Besides the paying customers, **292 signers paid their own share** of a signature, under a model the old platform did not have.

SINCE RELAUNCH **5,568 contracts**

81.9% completed by every signer; a median of 7.5 minutes from creating a contract to its last signature.

IN TOTAL **9,636 contracts · 7,576 users**

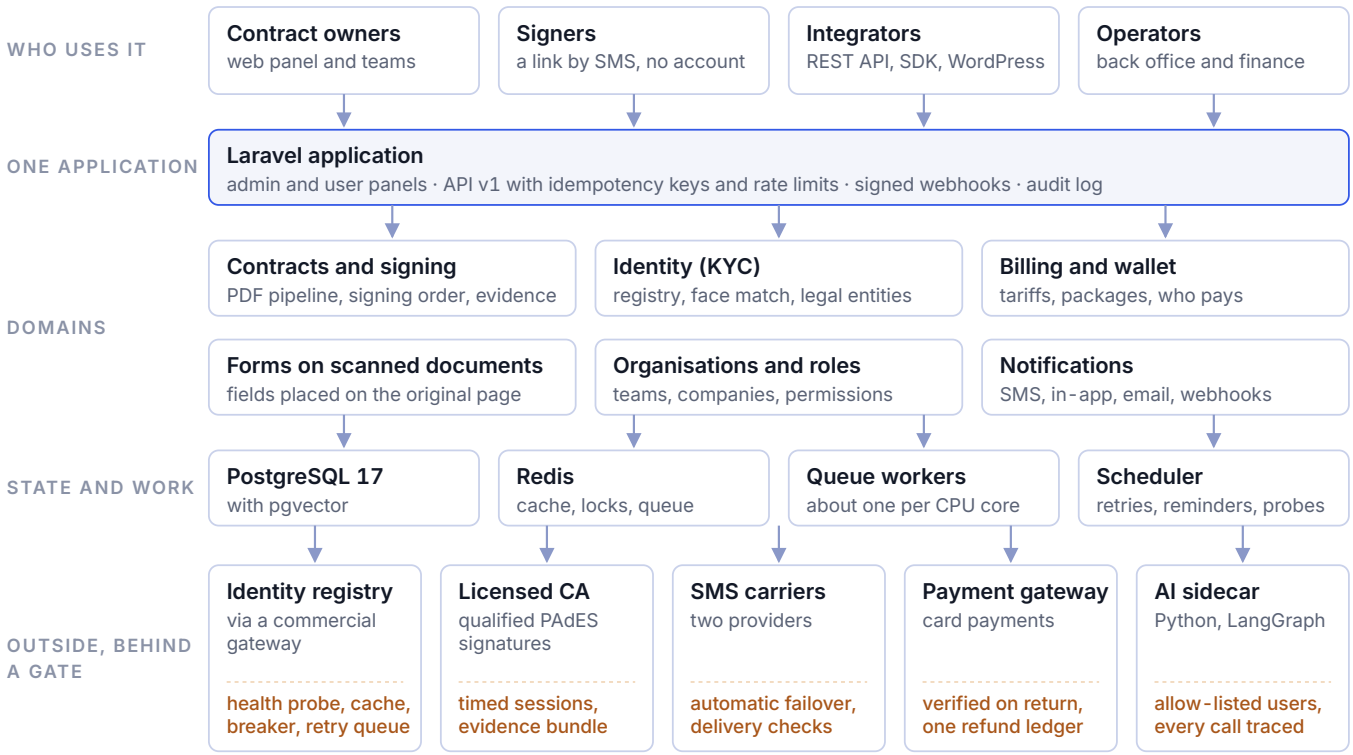
8,433 signatures from 6,422 distinct people, 198 of them qualified PKI signatures (since 11 June).

MONTH ON MONTH **five rises in a row**

550, 840, 1,036, 1,347 and 1,747 contracts from May to September: a median rise of 30% a month, with September counted at 25 days.

02 One application, many hard edges

A deliberately plain core, a Laravel monolith with one database and one queue, so that the difficulty lives where it belongs: at the edges, where identity, signatures, messages, money and AI meet the outside world. Each of those edges sits behind a gate with a defined way to degrade.



01

Plain core, hard edges

No microservices for their own sake. One team-sized codebase is easier to govern, test and roll back; the only separate service is the AI sidecar, because it has a different runtime and a different risk.

02

Every dependency behind a gate

Each outside provider has a health signal and a degraded mode, so an outage becomes a delay the user is told about, not a lost contract.

03

Slow work goes to the queue

Rasterising PDFs, signing, identity retries and notifications run in workers, and web requests stay short. The load lab showed the ceiling is CPU, not worker count: more workers than cores made it worse.

04

The API is a product

Idempotency keys against double submission, tiered rate limits, signed webhooks with retries and a dead-letter state, a PHP SDK and a WordPress plugin, so an integrator never has to guess.

MEASURED CEILING **2,556 contracts an hour**

through the API with one-page documents; 1,329 signed forms an hour. Load lab on a production-identical stack, 4 cores, 28 August 2026.

REAL LOAD **64.8 contracts a day**

average over the last 30 days, about 2.7 an hour; the busiest single hour this year had 27. Headroom is two orders of magnitude.

AVAILABILITY **99.98% of 1.04M**

external requests answered without a server error (162 errors), 25 Aug to 25 Sep 2026, from the web server's own logs.

03 When the identity registry drops

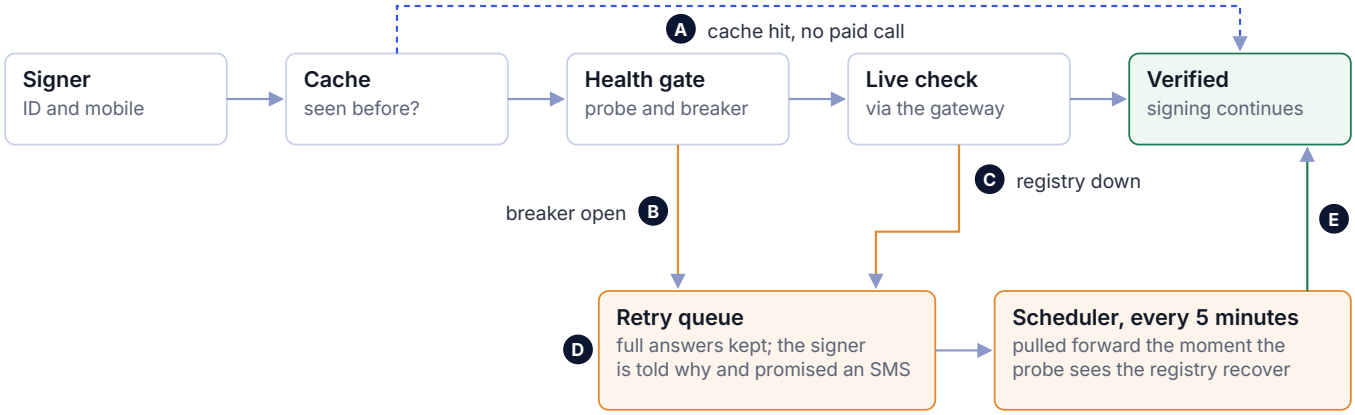
Before anyone may sign, the platform must confirm that the mobile number belongs to the person and that the national ID is real. That check goes through a commercial gateway to the national registry, which drops for seconds or minutes on most days: on 36 of the last 42 days in the log.

BEFORE A 20-second blip could destroy a contract

Contract processing retried once, ten seconds later. One policy served two unrelated failures: a broken file, where retrying is pointless, and a registry outage, where retrying is the whole point. The evidence was a natural experiment: two identical uploads, sixteen minutes apart, hit the same outage. One died on its single retry and one survived, and nothing distinguished them except which side of the blip the retry landed on. The owner of the first had to upload everything again.

AFTER Classify the failure, then treat it for what it is

A repeat lookup is answered from a cache and costs nothing. A health gate, fed by a probe and a circuit breaker, stops the platform from making signers wait on a registry that is known to be down. A real outage no longer fails anything: the full answers go into a retry queue, the signer is told why and promised a text message, and a scheduler retries every five minutes, pulling the whole queue forward the moment the probe sees the registry recover.



A 45% of 10,632 identity lookups answered from cache, with no paid call (4,834)

B 130 calls short-circuited while the breaker was open, instead of a signer waiting on a dead registry

C 977 outage events from the registry, on 36 of the 42 logged days

D 90 identity checks queued since June; 76 recovered (84%), a median 9.3 minutes after the outage

E 71 contracts carried through an outage instead of failing; 60 of them completed

A JUDGEMENT CALL The shortcut we refused

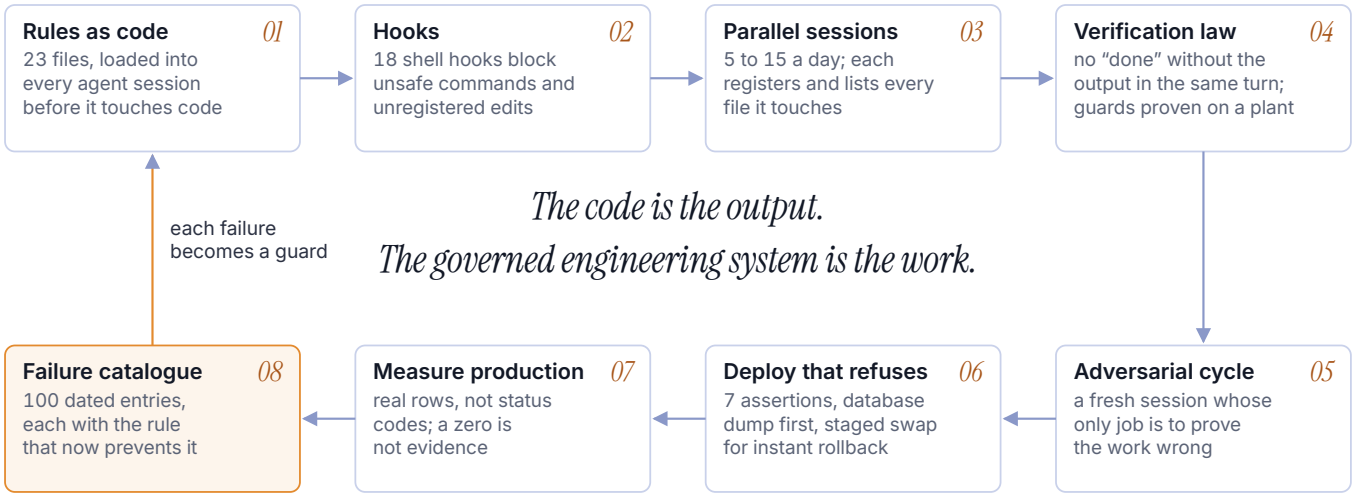
During an outage, verifying signers by a one-time SMS code only would have lifted completion immediately. It stays blocked. A signature that rests on a weaker identity check is a signature a court can question, and the whole value of the platform is that its signatures hold. Resilience here means waiting correctly, not lowering the bar.

STILL OPEN What it gets wrong today

The health gate reads a probe result cached for about five minutes. So the first signers in a fresh outage can still wait up to about sixteen seconds, four timed-out calls, before they are queued. The live failure streak that would close this gap already exists; reading it on the hot path is specified and not yet shipped.

04 The code is the output; the system that governs it is the work

An AI agent writes plausible code quickly and reports success confidently. Both are fine until the report is wrong. So nothing in this repository relies on an agent's diligence: every expectation is a file, and the important ones are machinery that runs whether or not anyone remembers them. Each part below is a file a reviewer can open.



RULES Doctrine, not style

The default answer to "deploy?" is no. Read the sources in full and find the mechanism before proposing a fix. A prohibition whose reason has expired is a defect to correct, not a wall to route around. 23 files, 9,401 lines.

THREE CYCLES The author never grades the author

Substantial work runs as three separate sessions: design, which must find a real defect in its own plan before it may close; build; and a fresh adversarial pass. On the reminder engine that pass found eight real defects the author had not seen.

DEPLOYS A pipeline that refuses

Manual dispatch with a typed confirmation. It will not move production backwards or onto a diverged branch, dumps the database before migrating, swaps a staged copy for instant rollback, and verifies with 7 independent assertions.

Open it yourself

PRIVATE REPOSITORY · OPEN UNDER NDA

FILE IN THE REPOSITORY	WHAT IT PROVES
`.claude/rules/` (23 files)	the doctrine every agent session loads before it may touch anything
`.claude/hooks/pre-bash-prod-guard.sh`	commands that could silently revert production are blocked, not discouraged
`bin/guard-exercise.sh`	every hook is fed a planted violation and must refuse it, in CI as well as locally
`bin/deploy-direction.sh` , `bin/deploy-verify.sh`	a deploy cannot move production backwards; seven checks decide whether it stays
`.claude/rules/self-inflicted-failures.md`	the 100 dated failures and the rule each one produced

What the system made safe to produce

GIT ON MAIN, SINCE 10 FEB 2026

4,373
commits since the first one on 10 February

2,751
commits in the busiest 30 days, from 8 August

~10,000
automated test methods in 1,328 test files

305K
lines of test code, against 323K of application code

318
database migrations, each run in a deploy that dumps first

100
dated failures in the catalogue, each with its guard

05 What broke, and the guard it became

Almost none of the 100 entries in the failure catalogue are coding errors. They are **measurement errors**: a check that could not fail, a search that matched a comment, a zero read as health, a number believed because it helped. That finding shaped the whole method. Three examples, told as they are written in the catalogue.

10 AUG 2026

Sixteen files became empty, and the syntax check passed them

WHAT HAPPENED

A code transform ran as a one-line command through a shell. The shell's escaping broke the pattern, the pattern failed to compile, the replace returned nothing, and the next line wrote that nothing to disk: sixteen source files, committed at zero bytes.

WHY NOTHING CAUGHT IT

An empty file has no syntax errors, so the checker passed it. The class then reported as "not found", which looks like a naming problem, not a deleted file.

THE GUARD NOW

Transforms live in files, never in one-liners. Before writing, the result must be non-empty, still a PHP file, and not implausibly smaller. A scan for zero-byte files runs after every bulk edit.

18 AUG 2026

A green test on a change that did nothing

WHAT HAPPENED

A layout setting was added to fix a crowded screen, a test asserted the setting, the suite passed, the change was deployed, and the screen looked exactly the same. The owner sent the second screenshot.

WHY NOTHING CAUGHT IT

The framework reads that setting from a different class. The test checked the input, the line that had been written, so it could not fail however wrong the change was.

THE GUARD NOW

Assert the output, never the setting: the rendered markup of every affected page. Before trusting a framework setting, find the line that reads it. Then plant the defect back and watch the test turn red.

19 AUG 2026

The wrong number was mine

WHAT HAPPENED

Checking another session's production table, three rows matched and one did not. I tested two explanations against their query, both failed, and I reported a defect in their numbers. My query had simply left out a filter they had stated.

WHY NOTHING CAUGHT IT

Every hypothesis I tested was about their instrument, none about mine. And my number made their conclusion stronger, which is exactly the kind of correction nobody checks.

THE GUARD NOW

Before calling a number wrong, reproduce its stated definition exactly. Keep "my query is wrong" on the list of hypotheses. When three of four rows match, suspect the data in the fourth, not the arithmetic.

The pattern, as four rules

DISTILLED FROM THE CATALOGUE

01

A guard must be seen to fail

Every check ships with a planted defect that proves it fires. A guard that cannot fail looks exactly like one that passed.

02

Assert the output

The rendered page, the stored row, the sent message. Never the setting, the call or the status code.

03

A zero is not evidence

Ask what the number looks like when nobody used the feature, and find one known-true case before trusting a search.

04

The author never grades the author

A separate, adversarial pass judges the work, because the mind that made a plan will also make its justification.

06 Business design, turned into systems

Who pays for a signature LIVE

The old model charged the sender for everyone. The new one lets the sender pay, split the cost with the signers, or name exactly who pays, with every refund going through a single ledger. It shipped behind a kill switch. The first counterparty payment arrived on 6 September; by the 25th, **292 signers had paid their own share**, and 24% of September's contracts used the mode.

Every signer who pays is also a person who has now used the product end to end: the split is a pricing model and an acquisition channel at once. The adversarial pass on it found a cancellation path that refunded the sender's share 150%, reproduced with a mutation test.

Reaching people LIVE

28,763 text messages since May (messages, not billing pages), 23,395 with a confirmed delivery receipt, over two carriers with automatic failover: 932 messages were re-routed when the first carrier failed. **9,602 in-app notifications** to 3,825 people.

Reminders run as one engine with per-person caps: 400 sent, 293 deliberately held back. The first fires on day two, because 95% of those who sign do so within two days (measured 27 August).

A multi-agent legal assistant

INTERNAL PILOT

A separate Python service built on LangGraph drafts contracts as a graph of agents: a **planner**, a **drafter**, a **citation checker** that verifies every cited article against a corpus of 3,738 statutory articles and sends a failing draft back for one redraft, a **critic** that does not loop back (automatic fixing amplifies errors), and an **adversarial reviewer**. Every model call is traced with its prompt version, tokens, latency, cost and a judge score.

Status, stated plainly PILOT

It is deployed and enabled for two internal users; no customer can reach it. Its adversarial review found that the citation checker looked laws up under different identifiers from the corpus, so it marked correct citations as fabricated. The fix is in the repository and waits for the service image to be redeployed. I would rather show a pilot with its defect ledger than a demo without one.

The rest of the surface

MEASURED IN PRODUCTION

198
qualified PAdES signatures via a licensed CA since 11 June; 118 certificates issued

5,932
public API calls from 21 keys since 3 July; 99.80% without a server error

324
signed webhooks delivered; 125 dead-lettered after retries when a customer's endpoint stayed down

25
companies as legal entities; identity, board and signing authority looked up in the official registry

Preparing for 20,000 signed forms a day

SCALE AUDIT, 30 AUGUST

A prospective integration raised the question of 10,000 to 20,000 signed forms a day. The company was ready to scale the hardware, so the audit had one test for every finding: **would more CPU fix it? If yes, drop it.** What is left is the work hardware cannot do.

Nine research lines produced **75 findings, 32 of them critical**, 16 re-verified by hand on the code or the live server. The worst were not about speed: a duplicate-submission guard that switches itself off silently, a per-customer billing lock that serialises paid submissions, and a queue whose health check could not see it had stopped.

What was measured versus what is fixed MOSTLY OPEN

The first money and safety fixes are built and waiting to deploy; most findings are still open. One finding was withdrawn during the build: reading the code in full showed its planned fix would have locked a legitimate customer out of their own account. The audit is a plan with evidence, not a claim of readiness.

07 What this means for you

You get two things: someone who can take an ambiguous, high-stakes system from understanding to production, and a method that makes AI-assisted building safe to trust. The method is portable. Its rules, hooks, verification protocol and failure catalogue are ordinary files, written so that a reviewer who has never met me can audit them.

Idea to production

8 TO 12 WEEKS

A goal and its constraints in; a live, governed system out, with the specification, architecture and failure modes written down on the way.

A product that outgrew its design

REBUILD FROM THE CONTRACT

Specify what the business, the law and the data require, move the data with every identifier intact, and switch over in minutes rather than weekends.

Technical leadership

FRACTIONAL CTO

Install the governance system in your codebase and your team, then run the adversarial reviews and deploy discipline with you until it runs without me.

Limits, stated before you find them

- **The growth is real and narrow.** Ten accounts created 89% of September's contracts, out of 66 that created any. Broadening the base is the next business problem; counterparty-pays is one lever, because every paying signer has used the product.
- **Two systems are not finished.** The legal assistant is an internal pilot, and most of the scale audit is open work.
- **One known resilience gap**: the first signers in a fresh registry outage can wait about sixteen seconds before being queued.
- **One human owner.** The git history has a single human identity. The mitigation is that the method lives in files anyone can read, not in my head.
- **Security certification is in progress**, not granted.
- **Self-reported, not provable from the data**: that work began in January 2026, and that the old code was never opened.

Methods and sources

EVERY FIGURE IS DATED

Production. Read-only SQL on the production database, 26 Sep 2026, 04:32 Tehran; no cache was cleared and nothing was written. September means 1 to 25 September (before 26 September 00:00 Tehran). "Old best" is the old platform's best calendar month for that measure between May 2024 and the relaunch on 18 April 2026: contracts 372 (August 2025), new users 298, paying customers 16. Money is compared as a multiple of the old best month (April 2025) and never shown in currency.

Paying customers are distinct users with a completed wallet top-up that month; counterparty payers are counted separately. **Completion** is contracts in the completed state among those created since 18 April 2026 22:18; the median is over completed contracts.

Identity. The lookup log starts on 16 August 2026, so cache and outage figures cover 16 August to 26 September. The retry queue dates from 14 June. A contract is "carried through" when a signer's check was queued and later succeeded.

Messages. Text messages are rows in the send log since 10 May 2026, counted as messages, not the billing pages a long message splits into. "95% sign within two days" was measured on 27 August.

Availability is from the web server's archived access logs, 25 August to 25 September 2026 (32 days; the archive keeps about 30). Local health probes are excluded; an error is any 5xx status. **Capacity** is from a load lab on an isolated stack built from the production image, 28 August 2026.

Engineering figures are from git on main at 56e966b1, 26 September 2026. Test methods count test-prefixed functions, test attributes and test annotations. Lines are PHP files only. The failure catalogue is the file that lists them, with dates.

Give me the goal. The specification, architecture and edge cases are my job.

Arash Banaeian
arash.bne@gmail.com · elemza.com/Arash
linkedin.com/in/arash-banaeian · Remote, UTC+3:30