CASE STUDY · ELEMZA · OCTOBER 2026

One engineer rebuilt a regulated signature platform in 70 days, *and the system that makes AI-written code safe to sign with*

eIEMZA verifies identities and carries legally binding signatures for Iranian businesses. This is how it was rebuilt from a one-page brief, what it carries today, what broke on the way, and the engineering method that let one person run a team's worth of AI coding agents without losing a contract.

Arash Banaeian
Systems Architect · AI-native SaaS · Fractional CTO
Sole engineer and technical owner of eIEMZA since January 2026

arash.bne@gmail.com
elemza.com/arash

5.8×
contracts in September against the old platform's best full month
2,167 against 372, before any marketing

914K
lines of code since 10 February, written by AI agents under one engineer's rules
308K of them tests

0
server errors from the public API in the last 30 days
4,833 calls · p50 40 ms · p99 127 ms

99.98%
of 491K requests served without a server error
4–30 September 2026

THE ARGUMENT

In January 2026 the old platform had all but stopped. 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. It went live on 21 April, 70 days after the first commit, moved two years of data in a two-minute cut-over with no contract lost, and in September carried **5.8 times** the old platform's best month.

I designed the architecture and governed the AI coding agents that wrote it. The engineering work is the system around them: rules every agent loads, hooks that enforce them, a separate adversarial review of every substantial change, a deploy pipeline that refuses, a load lab that found the limits before customers did, 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 where it still falls short.

READ IT IN TEN MINUTES

01 The situation and the result p. 2
02 Architecture: a plain core, hard edges p. 3
03 When the identity registry drops p. 4
04 Governing AI coding agents p. 5
05 What broke, and the guard it became p. 6
06 Scale, measured instead of assumed p. 7
07 A product and a developer platform p. 8
08 What this means for you p. 9

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 on most days.
- Sanctions and network filtering block part of the software supply chain; builds run through mirrors.
- 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 a reviewer can open. The requirements annex had 65 items; 19 complete subsystems were built that it never asked for.

From a one-page brief to five times the old best

TIMELINE

Jan 2026 The old platform is down to a trickle. 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.

30 Sep 2,167 contracts in the month: 5.8 times the old platform's best.

The result, before any marketing

MONTHLY, MEASURED IN PRODUCTION

	OLD BEST	SEP 2026	×
Contracts	372	**2,167**	**5.8×**
New users	298	**1,725**	**5.8×**
Money collected	—	**—**	**4.5×**
Median time to fully signed	14 min	**8 min**	**faster**
90% of contracts fully signed within	15 h	**8.7 h**	**faster**

Old best is the old platform's best calendar month for each measure, May 2024 to April 2026. Money is shown only as a multiple. Signing times compare both systems on one migrated dataset.

CUSTOMERS, NOT ONE BIG ACCOUNT

Monthly sending customers rose from **16 in May to 74 in September**. 223 customers have sent a contract; 57 of them five or more.

SINCE RELAUNCH

5,987 contracts, four in five completed by every signer (82.4%), a median of 7.8 minutes from creation to last signature.

IN TOTAL

8,819 signatures from 6,713 people; 215 of them qualified PKI signatures, 126 certificates issued. Monthly rises: 53%, 23%, 30%, 61%.

02 A plain core, and the difficulty where it belongs: at the edges

A Laravel monolith with one database and one queue, so that the hard problems live at the edges, where identity, signatures, messages, money and AI meet the outside world. Each edge sits behind a gate with a defined way to degrade.

WHO USES IT	**Contract owners** web panel and teams	**Signers** a link by SMS, no account	**Integrators** REST API, SDK, WordPress	**Operators** back office and finance
ONE APPLICATION	**Laravel 12 application** user, admin and writer panels · API v1 with idempotency keys and rate limits · signed webhooks · audit log of 46,268 events			
DOMAINS	**Contracts and signing** PDF pipeline, order, evidence	**Identity (KYC)** registry, face match, entities	**Billing** tariffs, wallet, who pays	**Forms** on scanned originals	**Notifications** SMS, in-app, email, webhooks
STATE AND WORK	**PostgreSQL 17** with pgvector	**Redis 7** cache, locks, queue	**40 job types** on 2 workers	**Scheduler** retries, reminders, probes
OUTSIDE, BEHIND A GATE	**Identity provider** 4 lookup types · cache, breaker, retry queue	**Licensed CA** qualified PAdES · evidence bundle	**2 SMS operators** automatic failover	**2 payment gateways** verified on return	**AI service** Python, offline embeddings

01
Plain core, hard edges
No microservices for their own sake. One codebase is easier to govern, test and roll back; the only separate service is the AI one, because it has a different runtime and a different risk.

02
Every dependency behind a gate
Each of the 7 outside providers 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
PDFs, signing, identity retries and notifications run in workers; web requests stay short. The load lab showed the ceiling is CPU, not worker count.

04
The API is a product
Idempotency keys, tiered rate limits, signed webhooks with retries and a delivery log, a cost-estimate endpoint, a sandbox, an SDK and a WordPress plugin, so an integrator never guesses.

What one 4-core server carries

PRODUCTION, MEASURED

~5%
CPU on one 4-vCPU, 8 GB server with HDD, running everything
~12% RAM · host up 126 days without a reboot

2,556/h
API contracts sustained in the load lab, about 61,000 a day
~800× today's volume, same server

40 ms
median public-API response; p99 127 ms; web median 62 ms
0 API server errors in 30 days

99.98%
of 491K requests without a server error, 4–30 Sep
74 server errors · status page: 99.80% over 90 days

By the numbers: 412 routes · 150 data models · 417 service classes · 87 controllers · 46 live UI components · 40 background job types · 320 migrations · 172 tables · 28 direct PHP packages, hash-locked. The status page counts a day with no traffic as "no data", never as 100%.

03 When the identity registry drops

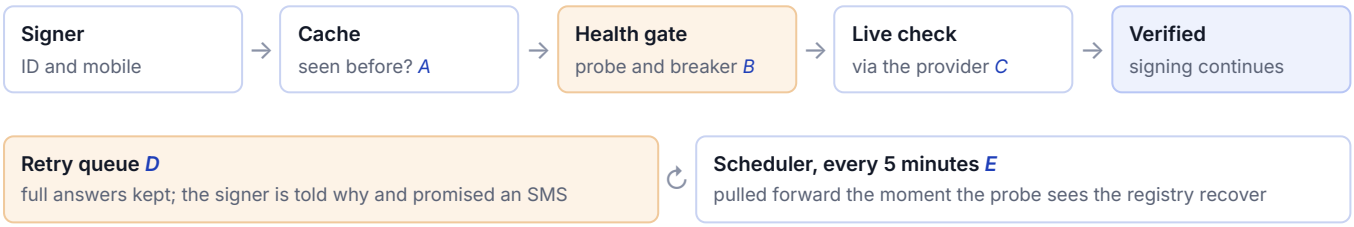
Before anyone may sign, the platform must confirm that the mobile number belongs to the person and that the national ID is real. That check goes to a national registry that drops for seconds or minutes on most days: on 41 of the last 46 days in the log.

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. Two identical uploads, sixteen minutes apart, hit the same outage. One died on its single retry and one survived; nothing distinguished them except which side of the blip the retry landed on.

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 signers from waiting on a registry 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 a scheduler retries, pulling the whole queue forward the moment the registry recovers.



A **46%**
of 12,334 identity lookups answered from cache, with no paid call (5,637)
16 Aug – 30 Sep

B **147**
calls short-circuited while the breaker was open, instead of a signer waiting on a dead registry

C **1,052**
outage events from the registry, on 41 of 46 logged days

D **92%**
of identity checks queued in September recovered (45 of 49), a median 5.7 minutes later

E **41**
contracts carried through an outage in September instead of failing; 34 completed

Self-healing, measured

PRODUCTION

89 of 100
identity checks that failed first time recovered automatically; 48 needed more than one retry, exactly where a competitor loses the user
11 Sep 2026

4,158
paid lookups removed by the cache in 30 days; the saving grows with volume

516
text messages rescued by automatic failover to the second SMS operator

A JUDGEMENT CALL · THE SHORTCUT WE REFUSED

During an outage, verifying signers by a one-time SMS code alone would have lifted completion immediately. It stays blocked. A signature that rests on a weaker identity check is one 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 about sixteen seconds 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.

Rules as code *01*
23 documents loaded into every agent session before it touches code

Hooks *02*
18 hooks block destructive commands and unregistered edits, and refuse to stop while guards are red

Parallel sessions *03*
up to ~8 at once on one repo, each registered, in 105 isolated worktrees

Verification law *04*
no "done" without the command output in the same turn; every guard proven on a plant

Three cycles *05*
design, build and adversarial review are separate sessions: the author never grades the author

Review agents *06*
8 specialists: a security reviewer on every PHP change, a drift detector against decisions

Deploy that refuses *07*
never backwards, database dump first, atomic swap for instant rollback, 45 guard tests

Failure catalogue *08*
100 dated entries, each with the rule that now prevents it

"THE AGENTS WROTE THE CODE. WHAT DID YOU DO?"

I design the system that produces the code: the product decisions and specifications, the architecture and data model, the rules, guards and protocols every AI session must obey, the review gates and the deploy pipeline, and the final judgement on what ships. I direct parallel sessions in writing, review evidence rather than prose, and turn every failure into a rule the machine enforces.

Anyone can ask an AI to write code. I built the engineering system that makes AI-written code safe to run with real money and binding signatures, and I am accountable for every line that ships.

IT CATCHES REAL DEFECTS

On the reminder engine, the separate adversarial pass found **eight real defects** the author had not seen. On the who-pays model it found a cancellation path that refunded the sender's share 150%, reproduced with a mutation test before it could reach a customer. The rules themselves are doctrine, not style: the default answer to "deploy?" is no, and a prohibition whose reason has expired is a defect to correct, not a wall to route around.

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; independent 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, SINCE 10 FEB 2026

4,462
commits on main since 10 February

266
commits in a single day, 31 August

2,041
commits in August alone

~7,800
test methods; 7,443 in the last full run

308K
lines of tests against 324K of product code

100
dated failures, 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, 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 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. Find the line that reads a setting before trusting 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 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. 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.

THE SAME DISCIPLINE, APPLIED TO THIS DOCUMENT

While these figures were being refreshed, the web-server log turned out to write every request twice, in two formats. An earlier availability figure had therefore doubled both the requests and the errors; the percentage was right, the volume was not. Every request count in this edition counts each request once. A number that looks better is the one to check first.

06 Scale, measured instead of assumed

A prospective integration asked about 10,000 to 20,000 signed forms a day. Instead of an opinion, I built an isolated load lab on the production host itself: the exact production build on a separate database, every outside vendor (certificate authority, SMS, identity provider) redirected *by address* to stub containers so the expensive code paths really run, k6 driving realistic journeys, and a watchdog guarding the live site.

BREAKING POINT, ELECTRONIC-SIGNATURE PATH

SIGNATURES / H	WEB WORKERS	BACKLOG	LOAD	WHAT HAPPENS
2,375	2	4	1.55	comfortable
2,640	—	10	—	**the queue starts to fall behind**
2,912	8	71	5.90	users see nothing yet; PDFs arrive late
3,100	25 / 25	119	9.12	**the web layer saturates**
3,231	25 / 25	173	15.83	the watchdog stops the test

2,556/h
sustained ceiling, one-page API contract: about 61,000 a day
5-page: 1,188/h · signed form: 1,329/h

~2,400/h
predicted on paper before the test; measured 2,640/h, with the right shape

Three things the lab found before customers could

26–28 AUG 2026

01 · MORE WORKERS MADE IT SLOWER

Going from 2 to 8 queue workers raised throughput only 26%, and the server reached the same stress point sooner. The bottleneck is CPU, not concurrency, so the upgrade plan is to buy cores, not workers.

02 · A SLOW QUEUE BECOMES AN OUTAGE

At saturation, **84.9%** of all server time went to clients polling "is my PDF ready?", against 10.2% creating contracts. A background slowdown cascades into a front-end outage unless polling is shaped.

03 · THE SILENT FAILURE COMES FIRST

The queue breaks (~2,640/h) before the web layer (~3,100/h). Users see no error; their PDF just does not arrive. So queue depth, not error rate, is the alarm that matters.

SAFETY, MEASURED RATHER THAN ASSUMED

- zero real text messages sent, zero calls to the real certificate authority;
- production database unchanged, row counts verified before and after;
- the live site answered in 0.27 s at peak load; the watchdog stopped the test at load 15.83.

THE SCALE AUDIT · ONE QUESTION PER FINDING

Nine research lines, 75 findings, 32 critical, 16 re-verified by hand on the code and the live server. Each finding had to answer one question: *would more CPU fix it?* If yes, it was dropped. What remained is the work hardware cannot do: a duplicate-submission guard that switches itself off, a billing lock that serialises paid submissions, a queue whose health check could not see it had stopped. Most fixes are still open; the audit is a plan with evidence, not a claim of readiness.

Not load-tested: the qualified-signature (PKI) path; its ceiling is calculated, not measured. Today's volume is about 74 contracts a day, so the measured ceiling leaves roughly 800 times headroom on the same server.

07 A product and a developer platform, not just a backend

The brief asked for e-signatures. What shipped is the whole path a business needs, form to contract to identity to signature to invoice, with an API good enough that integrators never have to guess.

DEVELOPER PLATFORM · ELEMZA.COM/DOCS

A public API that documents itself

- **45 endpoints** in 11 groups; **21 webhook events**, every delivery HMAC-SHA256 signed, with retries and a delivery log the customer can inspect.
- Docs with **255 code samples**, an OpenAPI spec and a Postman collection; idempotency keys, a sandbox, a quota endpoint and a cost-estimate endpoint, so you know the price before you call.
- **Every form a customer builds gets its own API page**: its endpoint URLs, its exact field keys, a ready-to-paste JSON body and cURL, and its webhook events.
- Last 30 days: 4,833 calls, zero server errors, p50 40 ms.

FORMS

A form builder that reads paper

- **32 field types**, including formula, repeater, line items, signature and four live government lookups (mobile to national ID, civil registry, company registry, postal code).
- An **image-overlay designer** turns a photo or scan of a paper form into a live digital form, with fields placed on the original page. Real forms average 4.1 pages; the largest has 39.
- A filled form bridges straight into signing: the filler becomes a signer and the original page comes back as a PDF with the answers overlaid.

PLUGIN V0.7.0 · 16.8K LINES

WordPress and WooCommerce

Embed any form with a shortcode, a Gutenberg block or an Elementor widget. An **offline queue** keeps answers when the site's connection drops. A WooCommerce contract form is **auto-filled from the order** and asks only for what the order lacks. Spam protection without a CAPTCHA.

MESSAGING · SINCE MAY

Reaching people

30,959 text messages over two operators with automatic failover (**943** re-routed); 10,412 in-app notifications to 4,117 people; 23 notification classes and about 150 SMS templates. Reminders run as one engine with quiet hours and caps: 484 sent, 316 deliberately held back.

BUSINESS MODEL · LIVE

Who pays for a signature

The sender can pay, split the cost with signers, or name who pays, with every refund through one ledger. In September **394 signers** paid their own share (414 payments), and 25% of contracts used the mode. Each paying signer has now used the product end to end.

PRODUCT-LED GROWTH · MEASURED

A growth loop nobody has pushed yet

6,861 people have signed a contract someone else sent them. 18 of the 223 sending customers met elEMZA first as a signer, a median of 4 days before sending their own. Conversion is 0.26%: a measured, untapped channel about 30 times the customer base, before any marketing.

A MULTI-AGENT LEGAL ASSISTANT · INTERNAL PILOT

A separate Python service drafts contracts through a structured interview, then a three-angle automated review (critique, internal consistency, risk flags). Every legal citation is checked against a corpus of 3,738 statutory articles, so a draft cannot cite a law that does not exist. Semantic search runs on an offline multilingual embedding model inside the container: user text is not sent out to be embedded.

STATUS, STATED PLAINLY

Enabled for internal users only; no customer can reach it yet. Its own adversarial review found that the citation checker looked laws up under different identifiers from the corpus. I would rather show a pilot with its defect ledger than a demo without one.

08 What this means for you

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.

8 TO 12 WEEKS

Idea to production

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

REBUILD FROM THE CONTRACT

A product that outgrew its design

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.

GOVERNANCE INSTALL · FRACTIONAL CTO

Make your AI-assisted team safe

Install the rules, hooks, adversarial review and deploy discipline in your codebase, then run them with your team until they run without me.

HOW WE WOULD WORK · YOU CARRY NO RISK

- **Paid by milestone.** Each milestone ends with something running and verified, in your repository and your accounts.
- **Stop after any milestone** and keep everything built so far. No lock-in, no notice period.
- **Written-first.** A daily written update with evidence: the command output, the screenshot, the link. Decisions in writing; calls when they help, with a written summary after.
- **Evidence over adjectives.** Every claim in a report comes with the output that proves it, the same law this platform was built under.

OPEN WORK, STATED PLAINLY

- The legal assistant is an internal pilot.
- Most of the scale audit is open work; the first money and safety fixes are built.
- The national information-security certification (AFTA) is in preparation, not granted; it includes a vulnerability assessment by an accredited lab. No external penetration test yet.
- One known resilience gap: the first signers in a fresh registry outage can wait about sixteen seconds before queueing.
- No external uptime monitor: availability is reported as requests served, not as uptime.

Methods and sources

EVERY FIGURE IS DATED

Production. Read-only SQL on the production database (BEGIN READ ONLY … ROLLBACK), 6 Oct 2026. September means the full calendar month, Tehran time; totals are cut at 1 Oct. “Old best” is the old platform’s best calendar month for each measure, May 2024 to April 2026: contracts 372 (August 2025), new users 298; money only as a multiple (April 2025). Signing times compare both systems on one migrated dataset (30 Sep). The retry-queue, webhook and API logs prune old rows, so those lines are September-only or last-30-days.

Traffic. Web-server logs, 4–30 September (the archive keeps about 30 days), health probes and the server’s own calls excluded, each request counted once. API latency from the application request log. **Capacity** from the load lab on the production image, 26–28 August, one-page documents. **Engineering** figures from git on main and from the repository, 6 Oct 2026, vendored libraries excluded. Code size is evidence of scope, not of quality.

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