The Brilliant Employee · 04

The homework that was impossible

For days a review queue read as my delay. The system had eaten it, and no part of the pipeline could say so.

sgnk.ai/brilliant ↗

Sagnik Mitra

Why this exists

sgnk is a one-person studio. A language model writes a large share of the production code. You do not need your own system to get answers out of a model. You need one to find out when the answers were wrong, because wrong work reads exactly like right work and no prompt fixes that.

Three things from that week, none visible from inside the tool: a sizing step ignored on 796 of 816 tasks, two fixes I had written up as done that were never made, and a detector whose 1,112 catches were mostly one test string of my own.

THE SYSTEM, AS OF 5 AUGUST 2026, 00:39Z	
rules written	11 June, 55 days ago
ledger running	27 June, 39 days
tasks logged	2,959, across 3,397 rows
controls on tool calls	**6 registered, 3 of them armed**

You can rent the model. You cannot rent the part that tells you it was wrong.

Five parts, and where today sits

Five parts sit around the model. What changes from one piece of writing to the next is which of them is under the microscope.

the Gate	sizes a job before a model is picked
the Gambler	recommends which model gets the job
the Rails	refuse the irreversible, before it runs
the Judge	grades finished work pass or fail
the Ledger	at least one append-only line per task

Listed in the order they act on a task, not the order they were built.

Today is about the Judge, marked above. The other four are the machinery around it, and this post does not assume you know any of them.

This page is the map. Nothing here needs another post to make sense.

Five graded examples before anything counts

My automation system's work is scored by a Judge, another piece of software that grades him. I do not trust a grader I have never graded, so the rule is written down: the Judge counts for nothing until it agrees with me on at least five real, human-reviewed examples.

> **Plain words: HUMAN GOLD.** a verdict a human actually made on a real task. The calibration currency. Machine-derived rows do not qualify, and the tool's own help text says so.

The dashboard sat at 2 of 5 for days. I read it the obvious way: I was the bottleneck. The system's own status reports agreed, filing it under waiting on the human.

A requirement of five with a count of two looks like human delay. Check the queue before you blame the human.

Two correct components, one starved pipeline

The sampler is the tool that serves items for review. Its rule was sensible: if a task already carries any gold row, it has been judged, stop serving it. That rule is correct exactly as long as gold means what it claims to mean.

Meanwhile an analytics pass had been writing mechanical rederivations, machine-recomputed labels, into the same gold file. Not maliciously. It was doing its job. Every row it wrote made one more task look already reviewed.

> **ASIDE.** Actor-first accounting: the analytics pass ate the pool, the sampler enforced its rule. Both behaved exactly as written.

Two components, each locally correct, starved the pipeline between them.

How the queue ate itself

1. Analytics pass writes machine-derived rows into the gold file, 83 of 85 tasks tagged
2. Sampler retires any task carrying any gold row, per its rule
3. Sampling pool drains to 0
4. Sampler reports 0 unadjudicated. No error, no warning, anywhere
5. Dashboard holds at 2 of 5. The human gets blamed

```
# before: any gold row retires a task
done = {r.get("correlation_id")
        for r in gold_rows()}

# after: only a HUMAN verdict does
done = {r.get("correlation_id")
        for r in gold_rows()
        if r.get("source") == "human"}
```

Every step is defensible in isolation, so no single component could report the failure.

A zero from a pipeline is indistinguishable from a broken pipeline until something asks: zero out of what.

83 of 85, the centerpiece number

83

85

gold rows that were machine output wearing a gold sticker

Only two rows in the entire gold file were verdicts a human had made. The other 83 were mechanical rederivations, analytics the tool's own help text explicitly labels NOT human gold. The sampler could not tell the difference because nothing forced it to.

One notch worse: a second consumer, the report that decides when the Judge counts as calibrated, gated correctly on the human count, but the precision it would print once that count reached five was computed over all 85 pooled rows. The trigger was honest. The number behind it would have been mostly machine agreeing with itself.

The queue was not just blocked. The counterfeit was about to be counted.

The second lock on the door

Suppose the pool had never starved and an item reached me. The function written specifically to recover the evidence behind each label was wired into a different subcommand and nothing else. The review screen could not show the work.

WHAT A SERVED SAMPLE ACTUALLY SHOWED	
correlation id	present
timestamp	present
machine verdict, kind, weight	present
files, diffs, output	**absent**

A verdict with nothing under it. Grade that.

So the homework was impossible twice over. No items could be served, and a served item could not be graded. Two independent locks, zero alarms between them.

Test the review path end to end as the reviewer, not as the plumber.

The fix, proven by differential

PRE-FIX	POST-FIX
sampling pool 0	sampling pool 83
human gold count 2	human gold count still 2
errors raised 0	set equality both directions

Measured at the fix commit, 29 July 2026. The 83 rows returning to the pool are exactly the 83 mechanical rows, checked as set equality in both directions. And the human count stays at 2, because those two really are the only human verdicts. A fix that moved that number would have been a second bug. Run the same computation today and the pool is larger, because more machine labels have landed since.

> **ASIDE.** A differential means asking the same question before and after the fix and diffing the answers. Cheap, and it ends arguments.

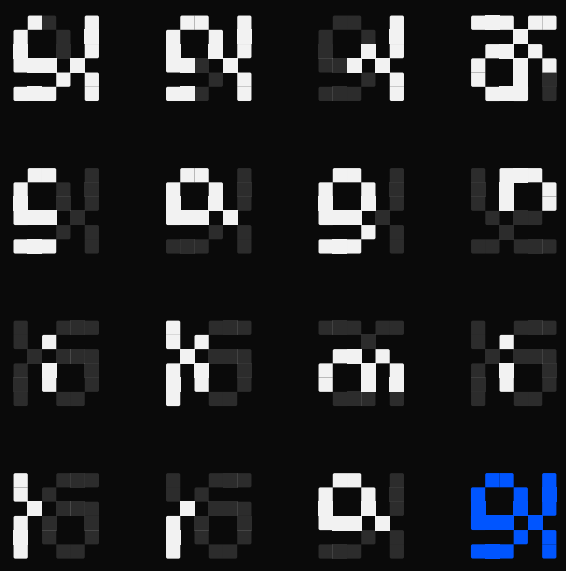
A good fix changes exactly the numbers it should, and proves it in both directions.

Zero out of what

The shape generalizes past review queues. A backup that writes nothing, a scanner that finds nothing, a queue that serves nothing: all print success. The only thing separating a healthy zero from a dead pipeline is the denominator, and who counted it.

1. Every zero gets a denominator before it gets believed
2. Every quiet system gets asked what it would look like broken
3. The starved pool is now a standing regression gate, gold-pool-not-starved, which had to prove it could fail before it was trusted to pass

Silent success is the worst failure shape there is. Make quiet systems prove they can be loud.



Nothing-to-review and nothing-needs-review print the same string. Ask zero out of what. Full dissection.

sgnk.ai/brilliant ↗

Sagnik Mitra

I build AI systems and the machinery that keeps them honest: gates, ledgers, judges, and rails. Product design for years, engineering the whole time. This series is what the building actually taught me.

`sgnk.ai · in/sagnikmitra · github.com/sagnikmitra`