The Brilliant Employee · 06

Failure arrives in the same tone as success

Sixteen nightly backup jobs, dead with exit code 78, and the room stayed perfectly quiet.

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 cuts across more than one of them, so no row is marked. Nothing below is assumed knowledge; everything this post needs, it explains.

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

Exit code 78. Sixteen jobs, every night, and no record of how long.

Nothing crashed. Nothing complained. No red banner, no failed-run email, no log line asking for attention. Sixteen scheduled backup jobs pointed at my project folders, and every single one died the instant it started, night after night.

The one scheduled job that lived in a plain hidden folder ran perfectly the whole time. That is precisely why I never went looking. A green light in one corner of the room reads, wrongly, as a green room.

> **Plain words: EXIT 78.** the code a macOS background job returns when the operating system refuses it before it can do, or say, anything

The absence of complaints is not the presence of backups.

The mechanism fits in two lines

macOS refuses background jobs access to Desktop, Documents and Downloads unless you explicitly grant Full Disk Access. A refused job dies before it can log a single word. That is the entire mechanism, it is documented, and I still lost weeks to it.

> **ASIDE.** For engineers: scheduled launchd agents hitting TCC-protected paths fail with getcwd errors and exit 78 on chdir. Keep scheduled scripts and their data on plain paths like ~/.sgnk or ~/Library, or grant Full Disk Access to a dedicated runner.

The macOS trivia is not the lesson. The lesson is that a safety system failed in a way that produced zero evidence, and every green assumption I held survived contact with nothing.

The interesting bug is never the mechanism. It is the silence around it.

Sixteen arrows stopped at an invisible wall

16 jobs into the project folders

stopped dead at the permission wall, exit 78, not one log line

the invisible OS permission wall

macOS guards Desktop, Documents and Downloads against background jobs

1 job in a plain hidden folder

no wall on its path, landed its output perfectly every night

The living job was my end-of-day archiver, parked under a hidden dot-folder the wall does not guard. Sixteen siblings a few path segments away hit the wall and vanished without a sound. I saw the survivor and generalized.

One perfect survivor is the best camouflage a dead fleet can have.

Dead and alive produce identical output

16 DEAD JOBS	1 LIVING JOB
crash **none**	crash **none**
alert **none**	alert **none**
log line **none**	log line **quiet success**
what I heard **silence**	what I heard **silence**

From where I sat, the two columns are the same column. A backup that ran and had nothing to report makes exactly the sound of a backup the OS killed on arrival. Failure arrived in the same tone as success, and it kept arriving nightly until I finally checked the dumps themselves.

If success and failure sound identical,
you are not monitoring, you are hoping.

Second specimen, same shape: the empty homework pile

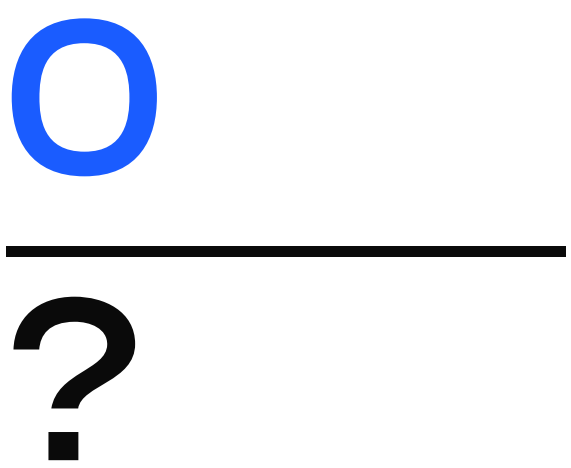
My automation system's finished work queues up for my human verdict. The count sat at 2 of a required 5 for days, and everyone, including me, read that as the human being slow.

WHAT THE QUIET QUEUE WAS HIDING	
reported	0 items awaiting review, no error
human verdicts	2 of a required 5, for days
rows in the pool	85
machine rederivations posing as gold	**83 of 85**
the sampler retired every item carrying any machine-written row: it ate its own pool	

Zero to review never meant the work was judged. It meant the pipeline had quietly made judging impossible, and it said so in the same tone as a job well done.

Nothing-to-review and nothing-can-be-reviewed print the same zero.

The question that breaks the spell: zero out of what



every quiet report is this fraction until something states the denominator

A checker that finds no problems and a checker that is broken emit the same message. The only defense is forcing the denominator into the report: scanned sixteen jobs, sixteen dumps landed, here are the byte counts. A zero with a denominator is information. A bare zero is a mood.

The probe I built does force a denominator. It prints how many repos it checked and how many of them failed, so a clean run reads as a positive count rather than an absence. Two other things I wrote on this page I never actually built: the report says nothing about how large the dumps are, and the sampler computes its pool size and then never prints it. I found both by opening the script while checking this page, which is the entire argument for opening the script.

Never accept a zero that does not say zero out of what.

Test the backup, not the intention

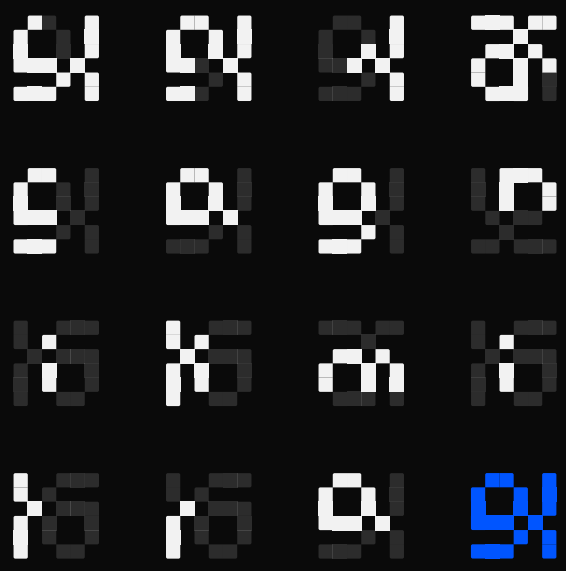
The schedule existing is an intention. The job firing is an intention. Only a restored file is a backup.

```
printf '%b' "[backup-probe]
  $bad failing /
  $(printf '%s' "$REPOS" | wc -w)
  probed\n$lines"
```

1. Every quiet job reports a denominator, not a feeling
2. A green light must be a positive signal, never the absence of a red one
3. Pull on the net on a schedule: open a dump, restore one table
4. Put scheduled work on paths the OS does not silently guard

This morning it used that honesty on me: two repos watched, two failing, the newest dump well past its threshold. The net is the only reason I know.

A safety net you have never pulled on is a rumor.



Silence is a report. Make every quiet system say zero out of what.

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`