SL SIMONE LABS

How Do I Know My Business Is Making Money?

What I found when I finally built something to check

Chris Simone

Simone Labs · 2026

SIMONE LABS

How Do I Know My Business Is Making Money?

What I found when I finally built something to check

Chris Simone · Simone Labs

2026

Written from the record of an operating business.

Contents

INTRODUCTION

The Question

Somebody asks you this at a party. A friend, a banker, your brother-in-law, doesn't matter. *So — is the restaurant making money?*

And you say yes. Because the doors are open, and Friday was busy, and there's money in the account. Twenty-three years in, that's not a lie. But it isn't an answer either. It's a feeling with a number stapled to it.

Here's the thing I'd never admit out loud until recently: for most of those years, my honest answer to "are you making money" was some blend of the bank balance, how the last few weekends felt, and a profit-and-loss statement I looked at weeks after the month it described. That P&L was produced by software. The software was fed by other software. The point-of-sale system told the bookkeeping system what we sold. The scheduling system told me what labor cost. Everybody's numbers came from somebody else's numbers, and at no point in that chain did anyone — including me — ever check.

I want to be precise about what I mean by "check," because this is the whole book.

I don't mean *look at*. I looked at those numbers constantly. I could tell you food cost and labor percentage off the top of my head on any given week. I mean check in the sense of: is the number the software is showing me actually the number? Does "net sales" mean what I think the words mean? When the automation that closes the books runs at six in the morning, does anybody find out if it didn't run at all?

I never asked. Why would I? I paid for the software. The software was the answer.

There's a version of this story I'm not going to tell you: the one where AI is magic. Where you plug something in and it finds hidden profit and you sit back and marvel. I've read those. They're written by people selling something, usually by people who've never had to make payroll or clean a grease trap. That story isn't what happened to me.

What happened to me was closer to an audit. A boring, grinding, one-thing-at-a-time audit that just happened to run automatically instead of once a year with an accountant looking backward through a shoebox. Every meaningful discovery in this book followed the same shape:

> **The software I already paid for was quietly wrong, and only building something else to cross-check it surfaced that.**

Not once did a machine tell me something I couldn't have known. It told me things I *could* have known and never would have, because catching them required somebody to sit down and compare two systems that both claimed to be right — every day, forever, without ever getting distracted or busy or tired. That isn't a job for a person. It's a job for a machine that never gets bored.

I found at least ten of these, depending on how you count the ones that came in pairs. Some had real money attached. Some had been making the business look *better* than it was, which is worse. One had been sitting there dead for two months without a single alarm going off.

By the end of it, the question in the title had changed for me. It stopped being "am I making money" and became something more useful:

> **How would I know if I weren't?**

That's the question this book is actually about. And the answer, it turns out, has almost nothing to do with artificial intelligence

and almost everything to do with verification.

A note on honesty, before we start

The build started in April 2026. The dated record in this book starts June 10, and I want to be plain about why those are different dates. June 10 is the first point where every fact I'm about to tell you could be pulled out of a log instead of out of my head. By then there were already something like twenty-two automated processes running for the pizzeria and the things around it, so April and May were not quiet months — they were just months I'd have to reconstruct from memory.

I'm not going to reconstruct them. In a book about the difference between believing a number and being able to prove one, memory doesn't get an exemption because it happens to be mine.

CHAPTER 1

A Baseline, and What Was Already Broken Inside It

I started building in April. By June of 2026, this was not a science project anymore.

There was a server running around twenty-two automated processes for the pizzeria and the things around it — bookkeeping, inventory, catering, email. It wasn't a pilot. It wasn't a proof of concept sitting in a folder. It was load-bearing. Real money moved through decisions those processes touched.

I want to pause on that, because most "AI for small business" material never gets past the demo stage. The demo is the easy part. Anybody can get something working once, on a Tuesday, while they're watching it. The hard part — the part nobody writes about — is what happens on day forty when you're not watching, and something breaks, and it breaks *quietly*.

Which brings us to June 10.

The bot that had been dead for a month

The bookkeeping automation had one job: post the monthly journal entries into QuickBooks. Processing fees from the point-of-sale system, the routine month-end adjustments. Unglamorous plumbing. The kind of thing that, when it's working, you completely forget exists.

On June 10 I found out it had been failing every single day since May 13th. My systems check caught it.

Almost a month. Every day it had tried, hit an authentication error, and stopped. Every day it had — supposedly — sent an alert about that. And every day that alert had itself been silently dropped, because an unrelated notification service had burned

through its sending quota and was quietly discarding messages.

So: a system failing, and the alarm about the failure also failing, and both of them failing in a way that produces no noise whatsoever. From where I sat, everything looked exactly like it does when everything is fine. That's the entire problem in one sentence.

The root cause turned out to be almost stupid. Two different machines were sharing one login credential, and every time one of them authenticated it invalidated the other one's session. They were stealing the key from each other, back and forth, forever. Fixed it the same day. Backfilled a month of bookkeeping in one catch-up run.

But I didn't really care about the fix. What stuck with me was the shape of the thing.

Automation that fails silently is worse than no automation. If I'd been doing those entries by hand, I'd have known on day one that I hadn't done them. The automation gave me something more dangerous than an undone task — it gave me *the confident feeling that it was done*. A month of false confidence. And the only reason I found out was that I looked.

That's the whole reason the rest of this book exists. I went looking one time, and what I found was so ordinary and so bad that I couldn't stop.

And then the revenue number was wrong

Five days later, June 15, I found a bigger one.

The script that pulled "net sales" out of the point-of-sale system had been including customer tips in the total.

Read that again if you run a restaurant, because you'll feel it in your stomach. Tips aren't sales. They don't belong in the top line, and counting them there inflates it by roughly three percent, every month, forever, in a way that looks completely plau-

sible.

Three percent doesn't sound like a catastrophe. Here's what three percent did.

I had been looking at a year-over-year sales comparison showing we were down 3.8%. Not great. Manageable. The kind of number where you say "soft year, we'll push through it" and keep doing what you're doing.

Corrected, the real number was **down 7.0%.**

The business was doing meaningfully worse than the dashboard said. Not better — worse. And I want to be honest about my first reaction, which was not intellectual curiosity. Realizing my revenue was miscalibrated was scary. I'd been making decisions off that number — not reckless decisions, just decisions with a little more confidence than the facts supported. Which is exactly how you end up in trouble slowly.

There's a thing people assume about business software errors: that they're random, so they'll wash out. Sometimes the number's high, sometimes it's low, no harm done.

That has not been my experience. Most of mine ran in the comfortable direction. Revenue looks a little better. Costs look a little lower. Not because anybody's cheating — because the defaults in these systems were chosen by somebody who wasn't thinking about my business, and "include everything in the total" is an easier default than "carefully exclude the money that isn't yours."

Not all of them, though. The labor number in Chapter 4 ran the other way — it made the business look worse than it actually was — and it survived just as long, because 29.8% is a believable labor number for a pizzeria. So what these errors have in common isn't direction. It's that they look plausible. A number that looks plausible doesn't get investigated. It just gets used.

So now I had two: a bot that had been dead for a month with no alarm, and a revenue figure that had been overstated the whole time. Two for two, on the first two things I looked at that month.

I stopped assuming the third one would be clean.

CHAPTER 2

Learning Not to Trust the First Answer

July is where I stopped treating these as isolated bugs and started treating them as a category.

The first thing I did was almost paranoid, and I'd do it again. On July 21, I built my own backup of the point-of-sale system's data — menus, modifiers, tax rates, pricing, the works — stored somewhere I control, independent of the vendor. Nobody asked me to. I did it because I'd started asking a question I'd never asked in twenty-three years of running the place: *if this company decided tomorrow that I no longer had access to my own history, what would I actually have?* The answer was: whatever they let me export, at whatever speed they felt like. That's not a criticism of the vendor — it's the deal everybody signs. But it's worth understanding that you signed it. Your history isn't a thing you own by default. It's a thing you have access to by permission.

I've come to think of that as the first real act of verification rather than a paranoia detour. You can't cross-check against a copy you don't have. Everything else in this book comes down to putting two versions of the same fact side by side — and one of those versions has to be yours.

The same bug, twice, and the second time it put a wrong number on the P&L for a month

Now the ugly one.

On July 23, I was troubleshooting something by hand — poking at the bookkeeping bot, the way you do — and I left a duplicate copy of it running in the background. Didn't notice. Nothing appeared to be wrong. The only symptom was some doubled en-

tries in a log file, which I didn't catch, because who reads log files when nothing looks broken.

That's mistake one. Mine, entirely. I want that clearly on the record, because I'm about to be hard on software vendors for the rest of this book and it would be dishonest to pretend I wasn't a source of errors myself.

On August 1, it happened again — and this time the timing did real damage to the numbers.

Two copies of the bookkeeping bot both woke up at 6 AM to close out July's books. They ran **100 seconds apart.** And in that 100 seconds, the underlying data pull was still filling in. So copy one closed July using an incomplete snapshot. Copy two closed July again, a hundred seconds later, using a *different* incomplete snapshot.

The result:

- July food sales overstated by roughly **$10,000**
- Reported net income overstated by roughly **$17,000**

For an entire month, the profit and loss statement for my business said we'd had a substantially better July than we actually had. It didn't look crazy. It didn't trip any alarm. It looked like a good month. If you'd handed me that P&L cold, I'd have believed it, because it was internally consistent — the sales supported the income, the income supported the sales. Everything agreed with everything else. It was just all wrong together.

It got caught by a cross-check comparing entries against each other, which found the duplicates and corrected them.

Here's the lesson I actually took, and it's not "be more careful."

I *was* being careful. I'm a careful person; you don't run one restaurant this long by being careless. The problem is that careful is something you are on a good day. It isn't something that runs at six every morning. You're careful right up until the mo-

ment you're tired, or it's Friday, or you got interrupted mid-task. A check that runs every day forever doesn't have good days and bad days. It doesn't get tired. That's the only real difference between it and me, and it's the entire difference.

Money going out the door that never went out the door

Then August 5, which is my favorite one, in a grim way.

I found an expense that had never been recorded. Not misclassified. Not in the wrong month. Never recorded, in any month, ever.

Third-party delivery platforms take a cut for running promotional programs — marketing fees. And here's the mechanism that made them invisible: **those fees are deducted before the money ever reaches the bank account.**

Think about what that does to conventional bookkeeping. Every system I had was fundamentally watching the bank. Money in, money out, categorize it, report it. That's how bookkeeping works for most small businesses, and it's a completely reasonable way to work — right up until you have a partner who nets out their own fees before they pay you. Then the expense has no fingerprint. There's no transaction to categorize. There's just a deposit that's slightly smaller than it should be, and nothing anywhere that tells you why.

When I finally found it and added it in, actual advertising spend for July turned out to be **roughly double** what the books had shown.

Double. Not off by a rounding error — off by half of the entire category. I had been spending real marketing dollars, month after month, with literally zero visibility. I couldn't evaluate whether that spend was working, because I didn't know it was happening.

And this is where I want to say something to any owner reading this who uses delivery platforms, or a payment processor that nets out fees, or any partner who pays you on a net basis:

Your books are only as complete as the money that touches your bank account. Anything a partner subtracts before paying you is, by default, invisible to your accounting. Not hidden. Not fraudulent. Just structurally invisible, unless somebody deliberately goes and gets it.

Nobody had gone and gotten it. For months.

CHAPTER 3

Food Cost, Rebuilt from a Silent Failure

If you'd asked me at any point in August what my food cost was, I'd have given you a number without hesitating. I've been quoting food cost from memory since I was in my twenties.

On August 20, I found out that the system generating that number had been dead for two months.

How it died

The failure had nothing to do with the pizzeria. The inventory and food-cost pipeline included a background process that called out to an AI model. At some point, the company that made that model retired that particular version — normal, happens constantly, entirely their right.

But when the version disappeared, the call started failing. And the failure wasn't loud. There was no crash, no red banner, no email. The process would try, get an error nobody was reading, and stop. Every automated food-cost update simply… didn't happen. Silently. For two months.

Same shape as the bookkeeping bot in June. Different cause, identical failure mode: **the system stopped doing its job and the absence of its output looked exactly like the presence of its output**, because nobody was checking whether the numbers were fresh. A stale number and a current number look the same on a screen. They're both just numbers.

That's twice now that the same category of problem cost me months. At that point it stopped being bad luck.

The rebuild, and what I did differently

I didn't patch it. I rebuilt it from the ground up on a much simpler design, and the design principle was this: **fewer things that can silently disagree.**

The old version had more moving pieces than it needed. The new one takes two inputs and only two: vendor invoices, and point-of-sale data. What I bought, and what I sold. That's it. Nothing else to keep in sync, nothing else to go stale, nothing else to quietly retire out from under me.

Then — and this is the part I'd underline for anybody building anything like this — I reconciled the new system against a manual physical count before I trusted it.

I went and counted. By hand. The old-fashioned way, the way I did it in 2003. And I compared what the new automated system said to what was actually on the shelf.

I'm belaboring this because it's the step everybody skips. New system produces a number, number looks reasonable, everybody moves on. But "reasonable" is exactly what every wrong number in this book looked like. The tips-in-revenue number looked reasonable. The double-closed July P&L looked reasonable. "Looks reasonable" isn't a check. The only thing that verifies a number is a completely independent measurement of the same reality, and for inventory, that's a guy with a clipboard in a walk-in cooler.

Automation doesn't get you out of counting. It gets you out of counting *every* time. You still have to count once, honestly, to earn the right to trust the machine after that.

The pizzas we were only half-counting

Four days later, August 24, I found something that had nothing to do with the outage — and had probably been wrong for a long time.

The point-of-sale system had two different paths a specialty pizza could be rung up through. Two legitimate ways for a staff member to enter the exact same product. And the reporting only ever looked at one of them.

Result: specialty pizza sales were being **undercounted by about a third.**

Not a third of one item. A third of an entire category — some of the best-selling things on the menu. For however long both paths had existed, which is longer than I want to think about.

Now, this one didn't cost me money directly. Sales are sales; the revenue hit the account whichever way it was rung in. What it corrupted was every decision downstream of "how much of this do we sell." Prep quantities. Par levels. Which items look worth keeping. Which specialty pies I'd have said were underperformers based on a report that was hiding a third of their volume.

Fixing it changed the actual prep quantities we needed for the busiest items on the menu. Real, physical, tomorrow-morning-different.

And I'll say the uncomfortable part: I'd been managing this menu for twenty-three years, and I did not catch, by feel, that a third of some items were missing from the report. My gut is good. My gut is not one-third good. Nobody's is. That's not a knock on experience — experience is what let me recognize the fix as correct once I saw it. But experience is a terrible instrument for detecting a consistent, proportional undercount. Consistent errors feel exactly like the truth. That's what makes them consistent.

CHAPTER 4

Labor Cost, Wrong in Both Directions

Food cost and labor cost are the two numbers every restaurant lives and dies on. Having just found out the first one was produced by a corpse, I went looking at the second.

Labor turned out to be worse. Not because the error was bigger, but because it was wrong in two different directions, from two unrelated causes, six days apart.

Direction one: hours that never happened

Also on August 20, I found out that a point-of-sale terminal used at lunch was never getting clocked out.

The mechanics: the terminal stays open, nobody closes the shift on it, and the system auto-closes it at 8 AM the following morning. So a five-hour lunch shift got recorded as roughly **seventeen hours of labor.**

Every single night.

Phantom hours, piling into the labor report, night after night, for who knows how long. And that phantom time was inflating the reported labor cost from a true **24.8%** to a false **29.8%.**

Five points. In this business, five points of labor is the difference between a good month and a bad one. That's not a rounding error, that's the whole argument.

Here's what got me: **the business had been quietly beating its own labor target the entire time.** I'd been reading a number that said we had a labor problem. We didn't have a labor problem. We had a *measurement* problem. The tool measuring labor was the thing that was broken.

Sit with the counterfactual for a second. If I'd trusted that 29.8% — if I'd taken it as fact and gone and made operational changes to fix a five-point labor overage that did not exist — I'd have been solving an imaginary problem with real consequences. Cutting into a business that was already hitting its target, and then wondering why service got worse and the number didn't move.

I don't think that's a hypothetical risk. I think it's one of the most common ways small businesses hurt themselves with bad data. Not by missing a problem. By confidently fixing one that isn't there.

Direction two: a cost estimate that couldn't do math on salary

Six days later, August 26, the same number was wrong the other way.

The scheduling software has a built-in "estimated cost" feature. You build the week's schedule, it tells you what that schedule will cost. Enormously useful, in theory. That's the number you'd naturally use to decide whether next week's schedule is affordable.

It was understating real labor cost by roughly **16%** — about **$868 in one week alone.**

Two reasons, both instructive.

First: **it cannot model a salaried employee at all.** Give it somebody paid a flat weekly rate and it doesn't know what to do, so it does something worse than nothing — it invents overtime hours for a person whose pay does not vary with hours. The tool isn't built for the concept of salary, so it produces confident nonsense instead of admitting it doesn't know.

Second: **its wage data silently goes stale.** Same word again. Silently. That's the third silent one in August alone.

Then it got worse. When I compared the point-of-sale system's wage data against the scheduling system's wage data, they **disagreed with each other** on multiple people's actual pay rates. Just — different. Two systems, both displaying a pay rate with total confidence, both used for real decisions, neither one right.

Only the payroll system itself was authoritative. Payroll's the one that actually cuts the check, so payroll is the one that knows.

So I built a small tool that does the obvious thing nobody's software will do for you: take actual recorded clock-ins, multiply by *confirmed* payroll rates, and report the real number. Explicitly bypassing both other systems' built-in cost estimates, because both of them are wrong and neither will tell you so.

The root cause, which is the actual lesson

When I traced why the wage data was wrong, I found something that explains most of this book:

> **A pay rate change has to be entered by hand into three separate systems. And the sync between those systems carries schedules and hours — but never wages.**

The systems talk to each other. They just don't talk about *that*. Hours flow. Schedules flow. Money doesn't.

Every wage error I found traced back to one of those three manual updates being missed. Not incompetence. Not carelessness. Just the completely ordinary reality that a human being has to remember to do the same thing in three places, forever, with no confirmation and no reminder, and eventually a human being will not.

That's the whole picture, and I think it's true of nearly every small business running more than two pieces of software:

Your systems are integrated in the places the vendors found easy to integrate, not in the places where being wrong costs you money. Nobody designed that. It's just where the seams landed. And the seams are where your numbers go bad.

Once I saw it that way, I stopped being angry at any individual tool. Not one of these systems is badly built. They're all fine at the job they were sold to do. The errors live in the gaps *between* them — and there is no vendor on earth whose job it is to look at the gaps. It's nobody's job. It's your job. And nobody tells you that when they sell you the software.

CHAPTER 5

Watching the Watchers

By August 21 I'd been through enough of these to see the pattern, and the pattern was not comforting. The bookkeeping bot: dead a month, no alarm. The food-cost pipeline: dead two months, no alarm. The notification service that was supposed to sound the alarms: quietly out of quota and discarding the alarms.

I had been thinking of each of these as an unfortunate exception. Lining them up, I finally understood I had it backwards. **A system failing loudly is the exception. Failing silently is the default.** Which meant the answer wasn't going to be "be more diligent." I'd already tried diligence. Diligence is what found these — after a month, and after two months. Diligence is *slow*.

The supervisor

So on August 21 I built a system whose only job is watching the other systems.

It doesn't do bookkeeping. It doesn't touch inventory. It has no responsibility for any business outcome whatsoever. Its entire purpose is to answer, continuously: **are the other things actually running, and do their numbers still make sense?**

It runs every five minutes. At launch it watched seven other automated systems, plus the website, plus the food-cost pipeline.

That's it. That's the whole idea. And I think it's the single most valuable thing I built in this entire stretch, which is funny, because it produces nothing. It sells nothing, saves nothing, calculates nothing. It just checks. Every five minutes, forever, without getting bored or distracted or having a busy Friday.

The math on it is straightforward. The bookkeeping failure went a month before discovery. The food-cost failure went two. With something checking every five minutes, that becomes minutes. The value isn't in catching a different set of problems — it's the same *kind* of problem, the dead-and-silent kind, caught two months earlier. The only thing it changes is how fast I find out. That turns out to be the whole thing.

Alert-only, on purpose

Here's the decision I'm proudest of, and it's a restraint, not a feature.

I launched it in **alert-only mode.** It tells a human. It does not act.

It absolutely could act. Restarting a dead process is easy — easier than detecting one. Every instinct in software says close the loop, make it self-healing, why would you page a person for something a machine can fix.

I said no, for a reason that comes straight from running a kitchen rather than writing code.

You don't hand somebody the keys on day one. The new guy can be talented, sharp, obviously going to be great — and he still doesn't close the store alone in week one. Not because you doubt him. Because trust is a thing that gets *earned through observed behavior*, and you haven't observed any yet.

I had just spent ten weeks finding out that just about every automated system I'd looked at was wrong in some way I hadn't anticipated. The idea that this new one — built by me, in the same conditions, with the same blind spots — would be the first one perfect enough to take unsupervised action on my production systems? That would have been an act of faith completely unsupported by the previous seventy-odd days of evidence.

So: it watched, it told me, I decided. Anything past that, it was going to have to earn.

The part that sat unresolved, and how it ended

The plan was for the supervisor to sit in alert-only mode until a review gate in late August, at which point I'd decide whether to let it act on problems automatically.

That date came and went. I never made the decision. Not "no" — *nothing*. It just kept doing what it had been doing, because the date passed and no alarm goes off about the absence of a decision. Which is, when I look at it honestly, the exact failure mode this whole book is about: something quietly stayed in a state nobody chose.

So I made the call. On September 6 I turned automatic remediation on, deliberately narrow: it can restart things, only things on a whitelist, with a cooldown between attempts and a hard daily cap so it can never sit there thrashing at something that isn't coming back. The catering agent is excluded on purpose, because that one is intentionally offline right now and a supervisor that keeps reviving something I switched off isn't help. Everything else it finds, it still just tells me about.

Two weeks later than it should have been, and it should have been a decision rather than a lapse. But it's a deliberate yes now, which is more than the date itself ever produced.

CHAPTER6

The Throughline — It Was Never the AI

First, what the AI actually did

Before I make that argument, I owe you an answer to the obvious question. If none of this was AI magic, then where exactly was the AI?

Here's the honest accounting, and it's smaller than you'd think.

Almost everything in this book is arithmetic. The cross-check that caught the doubled July entries compares entries against each other and flags the ones that appear twice. The labor tool takes recorded clock-ins, multiplies them by confirmed payroll rates, and adds it up. The rebuilt food-cost pipeline takes what I bought and what I sold and divides. The supervisor asks two questions — did this run, and do its numbers still make sense — every five minutes. None of that is intelligence. It's grade-school math, executed on a schedule, forever.

There is exactly one place in this record where a model was doing work at runtime, and it happens to be the one that broke. The old food-cost pipeline called out to an AI model as part of getting from a stack of vendor invoices to a food-cost number. When the provider retired that version, the whole pipeline went quiet for two months. The AI part is the part that failed.

So what was AI actually for?

This: I don't write code. Never have. Twenty-two automated processes, the cross-checks, the labor tool, a supervisor watching seventeen things every five minutes — none of it would exist if building it had required me to become a programmer first. That's the contribution, and it's enough. The AI didn't find the

errors. It made it possible for somebody like me to have the boring machinery that finds them.

The list

Let me put the main ones in one place, because the list is the argument.

- Revenue overstated, because customer tips were being counted as sales
- A month of books never closed at all — and then, weeks later, closed twice in the same 100 seconds
- Real advertising spend invisible, because it was netted out before the money ever hit the bank
- Food-cost tracking completely dead for two months, with no alert of any kind
- Best-selling menu items undercounted by roughly a third, because the register had two doors into the same product
- Labor cost wrong in *both* directions, at different times, from two entirely unrelated causes
- Three separate systems that each claim to know an employee's pay rate — and disagree with each other

Now read that list again and try to find the one where an AI told me something I could not otherwise have known.

There isn't one.

Every item on that list was **discoverable by any competent person who sat down and compared two systems that both claimed to be right.** No special insight. No prediction. No model doing something clever. Just: this number says X, that number says Y, why.

The reason nobody had done it is not that it's hard. It's that it's *relentless*. It has to happen every day, on every number, forever, and it has to happen most carefully on the days you're busiest —

which are precisely the days no human being is going to do it. The work isn't difficult. The work is *unrelenting*, and unrelenting is the one thing people are worst at and machines are best at.

That's the actual role all this machinery played. Not oracle. Not genius. **Tireless.**

What this actually is

The honest name for what I built is not "an AI transformation." It's an audit.

Not the once-a-year kind, where somebody experienced looks backward at a closed year and tells you what happened. The continuous kind. Running every day, on live numbers, while there's still time for the answer to change what you do.

Think about how strange the normal arrangement is. The most rigorous examination your business receives happens once a year, months after the fact, on numbers that are already history. The other three hundred and sixty-four days, you're flying on instruments nobody has calibrated. And you have no way of knowing whether those instruments are lying, because the only thing that could tell you is a second, independent instrument — which you don't have, because who buys two of everything.

That's what all of this was. Buying the second instrument.

So: how do I know my business is making money?

Here's my honest answer now, and it's less satisfying and much more useful than it used to be.

I don't know it because a number told me. I know it because the number survived being checked.

That's a different kind of knowing. The old kind was: the dashboard says X, therefore X. The new kind is: the dashboard says X, and an independent path to the same fact also says X, and

something has been comparing them every day, and it has not complained. The confidence doesn't come from the number. It comes from the *disagreement that didn't happen*.

I'd put it this way. I used to believe my numbers. Now I can prove them. Those are not the same thing, and for twenty-three years I could not have told you the difference.

What I'd tell another owner

If you take nothing else from this book, take these four.

1. **Every automated thing you own needs something else watching it.** Not because it's badly built. Because failure is silent by default and the absence of output looks exactly like the presence of output. If you can't answer "how would I find out if this stopped," it has already stopped at some point and you didn't find out.
2. **Errors survive by looking plausible — and most of them lean the comfortable way.** Revenue a little high, costs a little low. Nobody audits good news, and nobody audits a bad number that looks like every other bad month.
3. **The gaps between your systems are where the money hides.** Every vendor's tool works fine at the job it was sold to do. Nothing is wrong *inside* the point-of-sale system or *inside* the scheduling software. Everything is wrong in the seams — the fields that don't sync, the fees deducted before the deposit, the second path into the same menu item. Nobody's job is the seams. It's yours.
4. **Verify against physical reality at least once.** Count the walk-in. Compare against the payroll register that actually cut the check. A number that has only ever been checked against another number has never actually been checked. Somewhere in the chain, something has to touch the real world.

None of that is about artificial intelligence. All of it is about verification. What changed is who gets to do the verifying: **AI is giving operators the tools to really understand their own business — some for the first time.** I'm one of them. I ran this place for twenty-three years before I could prove a single number in it.

CLOSING

What I Do With This Now

I didn't set out to start a company doing this. I set out to find out whether my restaurant was making money.

But somewhere around the fifth or sixth of these — I think it was the delivery marketing fees, the money going out a door I didn't know existed — I started doing a different kind of arithmetic. Not about my business. About everybody else's.

Because here's what I know about the shops around me. They're running the same three or four pieces of software I was. A point-of-sale system. Something for scheduling. QuickBooks or the equivalent. Maybe a delivery platform netting out fees before the deposit. Same vendors. Same defaults. Same seams in the same places.

And almost none of them have anybody checking.

Not because they're careless — because it never occurs to you to check. It never occurred to me either. The software is supposed to be the answer. That's what you bought. Nobody sells you a point-of-sale system and mentions that its definition of "net sales" might not match yours, or that its wage data will drift out of sync with the other two places you have to type the same number.

So I'd bet — not on any specific shop, but on the category — that most owners reading this have at least one number on their dashboard right now that is wrong in a way that looks perfectly plausible, and no mechanism whatsoever that would ever tell them.

That's what Simone Labs is for.

The honest version of the offer

I'm going to describe this plainly, because a book that turns into a sales pitch in the last chapter is exactly the kind of thing I'd throw across the room.

I'm an operator, not a software vendor. I spent years paying for template tools — one product stretched across a thousand different businesses, never quite fitting mine. What changed recently is that it became possible to build the *specific* thing a shop needs instead of the general thing a vendor could sell to everyone. I did that for my own shop first. Everything in this book was built here, running on my money, breaking on my time.

The work goes in three steps, and they don't get skipped.

A fit call. Thirty minutes, free, and the honest outcome is sometimes "you don't need this." Some businesses have a front-door problem, not an operations problem — they need customers, not better numbers. I'd rather tell you that on a Tuesday than take your money for the wrong thing.

An operating assessment. You send the exports — point-of-sale, books, whatever the inbox looks like. I read a month of your real operation the same way I read mine, and I hand you a written map: where the numbers disagree with each other, where the seams are, what's worth automating and what absolutely isn't. The written map is yours either way. If you never hire me for anything else, you keep it.

Then, if it makes sense, the build. Custom systems on your accounts, doing your specific work — and, always, something watching them. The supervisor from Chapter 5 isn't an add-on. It's the part that makes the rest safe to rely on, and it's the part nobody else builds, because it's the part that doesn't demo well.

Two rules I don't bend on, both of them learned the hard way in these pages:

Nothing customer-facing goes out without a human's yes. The agent drafts, a person sends. Not as a limitation — as

the design.

It fails toward bothering you, never toward acting alone. Anything a system is allowed to do on its own is a short, pre-approved list with a hard ceiling on it. Everything else escalates to a person. It is never permitted to guess quietly, and it is never permitted to go silent. Every expensive thing in this book happened because something went silent.

I'm not going to tell you this fixes your business. It didn't fix mine. What it did was let me see mine accurately, which is a much smaller claim and a much more valuable one. Seeing accurately doesn't make a slow month into a good one — my corrected sales number was worse than my wrong one. It just means the decisions you make next are made against reality.

What I have instead of a testimonial is a system that's been running in a real, operating business for months, unprompted, whether or not anybody's watching — including this morning, while I was writing this. The bookkeeping ran. The food-cost pipeline is alive. The supervisor is doing its seventeen checks, and the two that are red are red for a reason I already know about and chose.

That's not a case study. That's just Tuesday. And after everything in this book, "just Tuesday" is the only proof I actually trust.

AFTERWORD

Honest Scope

A short note on what this book is and isn't, in the same spirit as everything else in it.

What's here: real, dated incidents from an operating business, June through early September 2026, drawn from project records rather than memory. The figures — 3.8% versus 7.0%, the $10,000 and $17,000 July overstatement, 29.8% versus 24.8% labor, the roughly 16% scheduling shortfall, the onethird undercount — are all from those records.

What isn't here: a long track record. This is months, not years. There are no customer testimonials, because I haven't earned any yet. There's no before-and-after profit chart, because that's not what this project was — it was about measurement accuracy, not a turnaround. And there's no April or May, because the build started in April and the trustworthy record starts June 10.

What's still open: the supervisor's automation decision from Chapter 5 is closed — made September 6, narrowly, and two weeks later than it should have been. What's open is everything that decision opens. A whitelist and a daily cap are my best guess at the right guardrails. I won't know whether they were until something fails in a way I didn't anticipate, which, based on everything in this book, it will.

And the one thing I genuinely can't tell you: how many of these are left. I found ten-plus of them over about three months. I don't believe I've found the last one. I don't believe there is a last one. Every one of them had been sitting inside software I was paying for, looking exactly like a correct number, for as long as nobody compared it to anything. There's no reason to think that stopped on the day I finished writing this.

That's not a sales pitch. That's just what "checking" turns out to mean.

• • •

Amore Pizzeria has operated in Zionsville, Indiana since 2003. Simone Labs is its author's AI practice, built on systems that ran in his own business first.