AI OPERATIONS · AUDIT FRAMEWORK

The AI Workflow Audit

How I run it, what you get, and a redacted look at the deliverable.

PREPARED FOR	PREPARED BY	VERSION
Overview for partners	Abdul Sami	2026-06-18

READY TO PLUG IN

A fixed audit framework and a designed report that comes out the same shape every time. This overview walks the process and shows a redacted slice of a real deliverable.

01 — Overview

What this is.

This is how I run an AI workflow audit, and what the finished report looks like. The point is simple: find where a business loses hours to manual work, where its tools overlap or fall short, and where AI actually earns its place. Then hand back a plan the owner can act on without me in the room.

The process is fixed, so the deliverable comes out the same shape every time. A plain executive summary. The one thing to fix first. What is already working. A prioritized roadmap with the value written next to each item. Every finding reads the same way: what I found, why it matters to the business, what to do about it.

The rest of this document is a redacted slice of a real audit, so you can see the format before anything is on the line.

4	1	3
review lenses, every audit	urgent fix on page one	week roadmap to hand off

02 — The deliverable, redacted

One thing first.

PAGE ONE OF EVERY REPORT

The single most urgent fix, before anything else.

Every report opens with the single most urgent fix, before the long list, so it never gets buried.

In a recent audit that was a set of live client API keys still recoverable from the project's history. One short paragraph, in plain language: what it is, why it matters, and the exact step to take that day. The other forty-nine findings were for later. This one was for that morning.

Area 01

Where the hours go

What I look for. The manual work that eats the week. The copy-paste between tools, the report someone rebuilds by hand every Monday, the data re-keyed from one system into another. I map it from a short intake and a look at how work actually moves, not the org chart.

Why it matters. This is where the hours come back. Most "we need AI" problems are really "this step is done by hand and shouldn't be."

Redacted example. A nightly job was regenerating the same summary ten times a day, whether the underlying data had changed or not. Quiet, invisible, pure waste. The fix was a schedule change and a check that skips the work when nothing moved.

Area 02

The tool stack

What I look for. What the business pays for, what overlaps, what is missing, and what is quietly expensive. Two tools doing one job. A subscription nobody opens. A workflow held together by one person's personal login.

Why it matters. The cheapest automation is often cancelling a tool or merging two, before any AI is involved.

Redacted example. A billing pipeline depended on one employee's personal access token. If that person left, the invoicing would have gone quietly stale. The fix was a shared service account, not a new tool.

Area 03

Where AI actually helps

What I look for. The few places AI earns its keep: drafting, summarizing, sorting, first-pass research. And, just as important, the places it does not, where a fixed rule or a small script is cheaper and more reliable.

Why it matters. Pointing AI at the wrong step burns money and trust. Half the value of the audit is telling you where not to use it.

Redacted example. A client wanted a chatbot. The real win was a weekly digest that wrote itself from data they already had. Smaller, cheaper, and used every week.

Area 04

Cost and risk

What I look for. Where spend can run away (uncapped AI calls, no per-tool visibility) and where the business is exposed (secrets in the wrong place, client data crossing borders without anyone deciding to).

Why it matters. A workflow that saves an hour but leaks a client key is not a saving. Cost and risk get checked on every audit, not just the happy path.

Redacted example. An AI feature had a cost field in its database that was never filled in, so nobody could answer "what did we spend yesterday." The fix made spend visible and added a hard cap.

Strengths

What is working well.

Every report has this section, and it is not filler. People act on a plan they trust, and they trust it more when it is honest about what is already good.

I name the things the team got right and tell them to keep those as the model for the rest of the work. An audit that only lists problems gets read once and filed. One that starts from what already works gets acted on.

Recommended priority sequence

A roadmap you can hand off.

The roadmap is sequenced, not a flat list, so the team knows what to do first.

Today. The one urgent fix from page one. Usually small, always done same-day.

Week one. The foundation. Visibility into spend, one source of truth for the data that matters, and the quick wins that pay back immediately.

Week two. The higher-leverage changes. The manual step that gets automated, the tool that gets cut, the workflow that gets one clear owner instead of none.

Week three. Hardening and cleanup. The things that are not urgent but rot if left alone.

Each item carries its value next to it, in plain terms, so the owner can decide what is worth doing without a technical translation.

Appendix A

Sample findings, redacted.

These are real findings from past audits, stripped of anything that could identify the client. They show the card format: a severity tag, where it lives, and the same three-part write-up every time.

F-1

Live API keys recoverable from version history

CRITICAL **Category:** Secret exposure **Location:** [redacted]

What I found. A set of live client API keys had been committed to the project's history, then deleted from the current files. Deleting a secret from the latest version does not remove it from the history. Anyone with a copy of the project could still recover working keys.

Why it matters. These were working credentials to client accounts. A single stray copy of the project meant those accounts had to be treated as compromised.

What to do. Rotate every key that day. Move them into environment variables. Rewrite the offending history if the project is ever shared outside the team.

F-2

AI spend defined but never measured

MEDIUM **Category:** Cost and observability **Location:** [redacted]

What I found. The system had a place to record the cost of each AI call, but nothing ever wrote to it. The most basic question, "what did we spend yesterday," could not be answered from their own data.

Why it matters. You cannot control a cost you cannot see. Spend had already spiked unpredictably more than once.

What to do. Compute the cost on each call from a simple per-model rate table, roll it up daily, and alert when it crosses a threshold. Add a hard cap so a stuck loop cannot run up the bill.

F-3

Admin actions left no trace

MEDIUM **Category:** Forensics and compliance **Location:** [redacted]

What I found. The system was built to record who did what (who changed a rate, who deleted a record), but the log was never actually written to. There was no way to answer "who did this, and when."

Why it matters. Without it, incident response and basic compliance are impossible, and a compromised admin account leaves no trace behind.

What to do. Write a small audit-log helper and call it from every sensitive action, recording who, what, and when. Surface it in the admin screen so it becomes a tool the team uses, not just a table nobody reads.

Appendix B

Method and scope.

Intake. A short questionnaire and one call to see the business, the tools it pays for, who does what, and where the time goes. Read access to the systems under review.

Review. I work through everything against the four lenses in this document: where the hours go, the tool stack, where AI helps, and cost and risk.

Findings. Each one is rated by impact and written the same way, in plain language: what I found, why it matters to the business, what to do.

Report. The designed document whose format you are reading now. Executive summary, the one thing to fix first, what is working, the prioritized roadmap, and the full findings list.

Follow-up. A call to walk it through. The team can implement from the report on their own, or I can.

A note on tools. I recommend the cheapest thing that does the job, not the one with the best demo. The build side is usually plain code; the recommendation side is tool-agnostic and picks whatever fits the budget and the team.

A note on writing. The report is written for the owner, not the engineer. If a finding cannot be explained in plain language, it is not finished.