FOR THE SOLO OPERATOR RUNNING AGENTS

The HTML Review *Loop.*

How I run a whole business off a team of agents from my phone. The one piece nobody hands you is the review loop, the moment the work comes back for your call. This is how I built mine, and the prompt to build yours.

READ TIME	TIERS	TOOLS NEEDED	BUILD TODAY
9 *min*	2	0	*Yes*

BY BOUBACAR

THE BOTTLENECK

You can run ten agents. You can only *read* so fast.

An agent can draft a cold email, a landing page, and three social posts before you finish your first task of the day. Then it stops and waits for you. That wait is the whole game. The agents do not slow you down. The reviewing does.

I run my business on a stack of agents I built myself. The work piles up faster than I can look at it. For a while my whole day was a scroll: open a file, read it, decide, type a reply, open the next one. Most of that motion was friction, not judgment. I was spending my attention on opening and closing things instead of on the one thing only I can do, which is decide whether it sounds like me and whether it is right.

So I changed how the work comes to me. Not the agents. The handoff. The agents got better the moment I stopped reviewing them like a stack of documents and started reviewing them like a control panel I carry in my pocket.

> ***The constraint in an agentic business is not the agents. It is the human throat the work has to pass through. This guide is about widening that throat without lowering the bar.***

What you will be able to do by the last page

Route every approval to the right surface, so a yes/no never costs more than a thumb tap and a real edit always carries the reason behind it. Review on your phone, on the move, with no app to install and no markdown to squint at. And hand the decision back to your agent in a form it can act on the same hour. The whole thing runs on plain HTML. You can hand the pattern to any agent stack you already use. The last section is the prompt to do exactly that.

THE CORE IDEA

Two tiers. Send each decision to the *cheapest* surface that can carry it.

Not every decision deserves a dashboard. A button is faster when a button is enough. The skill is knowing which is which.

TIER 1 / BINARY

The button push

Yes or no. Approve or reject. Nothing to explain. A message lands on your phone with the decision and two buttons. You tap one. Done.

This is a chat message, not a page. It belongs anywhere you already get notified.

Use when: the answer has no nuance and needs no note. Send it. Schedule it. Kill it.

TIER 2 / NUANCED

The HTML dashboard

An edit with a reason. A "yes, but change this one line." A judgment that only makes sense if you can say why. This needs room to type and the full content in front of you.

This is a single web page, built for your thumb, that you open from a link.

Use when: the verdict would need a sentence of explanation to be useful to the agent.

The one test that routes everything

Before sending any approval to a human, the agent asks itself a single question:

Would an "edit" verdict here need a sentence to explain it? If no, it is Tier 1, send a button. If yes, it is Tier 2, build a card on the dashboard.

That is the whole routing logic. It is small on purpose. A binary decision dressed up as a dashboard wastes your time. A nuanced decision crammed into a yes/no button loses the reason, and the reason is the part the agent actually needs.

The button captures the verdict. The dashboard captures the why. Most agent work dies for lack of the why.

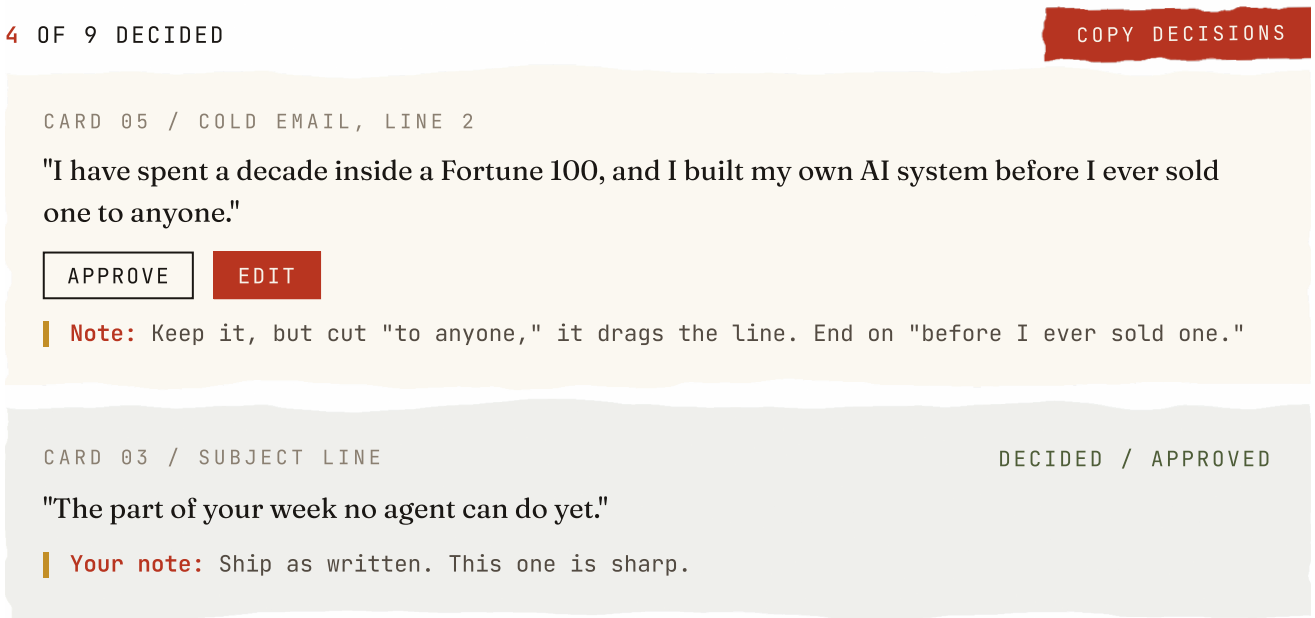
THE ROUTING RULE

WHAT THE PAGE IS MADE OF

Seven parts. Each one earned its place by *fixing* a real annoyance.

- **Full content inline.** The whole draft sits on the page. No "open to view," no link to a doc. You read the thing you are deciding on, right there.
- **Approve / Edit toggle per item.** Two states, one tap. Approve means ship as written. Edit means I am keeping it but changing something, and the note explains what.
- **A notes field that holds the why.** This is the heart of it. "Cut the second line, too salesy." "Swap Guinea for Conakry, more specific." The note is the instruction. The verdict alone is not enough.
- **Copy decisions.** One button compiles every open verdict and note into clean plain text. You copy it, paste it back to the agent, and it gets to work. That paste is the contract.
- **Locked decided cards.** Once you decide a card, it locks. Controls hide, a banner shows your verdict and note. A re-paste never resends something you already acted on.
- **Collapse toggles.** Long pages get unwieldy. Every card folds to a one-line summary. Decided cards default folded, open cards default expanded, so the work left to do is always what you see first.
- **A decided-of-total bar.** Sticky at the top. "4 of 9 decided." You always know how much is left, which is what lets you start a queue and finish it later without losing your place.

What it looks like on the glass



EVERY PART CAME FROM A REAL FRICTION

I did not design this. I *collided* with it.

It started as a row of buttons. Each piece I added came from a specific moment where the last version broke.

1 It started with buttons

Telegram, two buttons, approve or reject. Fast and clean for yes/no. But the day I wanted to say "yes, but fix one line," I had no way to say it. I typed a paragraph into the chat, the agent half-understood, and we went three rounds. The button could not carry an edit.

2 So the nuanced stuff moved to a page

I built one HTML page: the full content, an edit toggle, a notes box, and a copy button. Now the edit and its reason traveled together. The agent got "change this, because that" in one paste. The buttons stayed for the binary work. That split became the two-tier rule.

3 Then I caught myself re-sending decisions

I would copy the decisions, the agent would work, and an hour later I would reopen the page and copy again, accidentally resending verdicts it had already acted on. So decided cards now lock. The copy button only ever compiles the open items. A re-paste is always safe.

4 And one stale note taught me to version

The content on a card got updated, but my old note was still sitting there, and I pasted it back word for word against the new version. Wrong instruction, confidently sent. Now every card carries a version. Update the content and the old note archives itself instead of riding along. Cards also collapse, so a long queue stays readable.

None of this was planned. Every feature is a scar. That is why it works: each part is the fix for a failure I actually hit, not a feature I imagined someone might want.

THE FORMAT ARGUMENT

Markdown is great for writing. It is wrong for *deciding*.

Agents love to hand you a markdown file. It is the path of least resistance for them and the path of most friction for you.

On a phone
WHERE YOU ACTUALLY REVIEW

A markdown file opens in a code view or a viewer with pinch-zoom and sideways scroll. **HTML lays out for the screen it is on.** You read it like a page, not a file.

Making a decision
THE WHOLE POINT

Markdown is read-only. To respond you switch apps, retype, and lose context. **HTML carries the buttons, the notes box, and the copy-back inside the same page.** Read and decide without leaving.

Tooling required
FRICTION TAX

A rich markdown experience wants an editor, an extension, or an app. **HTML needs a browser, which every phone already has.** Zero install, open from a link.

Holding state
ACROSS A SESSION

A markdown file forgets you the moment you close it. **HTML remembers your verdicts in the browser,** so you can decide four, walk away, and finish the other five later.

The honest counter

Markdown wins for things meant to be written, diffed, and version-controlled: docs, code, notes between agents. Keep it there. The argument is narrow and specific. For the moment a human has to look at agent output and ***decide on it***, HTML is the better surface, because deciding is interactive and reading a file is not.

A deliverable you only read can be a file. A deliverable you have to act on should be a page.

WHY HTML FOR AGENT OUTPUT

HAND THIS TO YOUR OWN AGENT

You do not need my stack. You need the *pattern*.

This works with any agent that can save a file you can open, a coding assistant like Claude Code, Cursor, or similar. Copy the prompt below, paste it to yours, and point it at whatever it just made for you.

The loop is three moves, and your agent does most of them. **One,** when it finishes work that needs your call, it generates a review page instead of dumping a file. **Two,** you open the page on your phone, decide, and tap "copy decisions." **Three,** you paste that back, and it acts. Here is the prompt that sets it up.

Swap the styling for your own taste. The structure is what matters: full content, a verdict, a reason, a clean copy-back, and locked items so nothing gets re-sent. Build the smallest version first. One card, one toggle, one copy button, and add the rest the first time you feel the friction, the same way it grew for me.

No coding agent, just a chat assistant? Ask it for the HTML and save the block it gives you as a file named review.html, then open that. One caution on the come-back-later part: a page opened from inside a messaging app sometimes loses its saved state. If that bites you, have your agent bake your verdicts into the page each time it regenerates, so the state lives in the file, not the browser.

COPY AND PASTE TO YOUR AGENT

From now on, when you finish work that needs my approval, do not hand me a markdown file or a wall of chat. Build me a single self-contained HTML review page and give me the path to open it. Follow these rules:

ROUTING
- If a decision is pure yes/no with no nuance, just ask me in chat. No page.
- If an "edit" verdict would need me to explain why, it goes on the page.

THE PAGE
- One HTML file, no external dependencies, mobile-first (readable on a phone).
- One card per item. Each card shows the FULL content inline, not a link.
- Each card has: an Approve / Edit toggle, and a notes textarea for my reason.
- A sticky bar at the top: "N of M decided."
- A "Copy decisions" button that compiles ONLY the still-open items into clean plain text as {item id, verdict, my note} and puts it on my clipboard.
- Persist my verdicts and notes in localStorage so I can decide some now and finish later without losing state.
- When I decide a card, LOCK it: hide its controls, show a banner with my verdict and note. "Copy decisions" must never re-send a locked item.
- Give every card a version id. If you update a card's content later, archive my old note instead of carrying it onto the new version.
- Long pages: let each card collapse to a one-line summary. Decided cards start collapsed, open cards start expanded.

THE LOOP
- After I paste the decisions back, act on each one the same session, and regenerate the page with the acted items locked.

Keep the page plain and fast. The job is for me to read, decide, and hand the decision back to you with as few taps as possible.

BEFORE YOU CALL IT BUILT

The page works when all of these are *true.*

- ☐ **I can read every item in full** without opening a link or zooming.
- ☐ **Each item has a verdict and a place for the reason** behind an edit.
- ☐ **One button hands the decisions back** as clean text I can paste.
- ☐ **It only ever sends the open items,** never the ones I already decided.
- ☐ **Decided items are locked** and show what I chose.
- ☐ **I can stop halfway and come back** with my decisions still there.
- ☐ **It reads cleanly on my phone** with my thumb, in one hand, on the move.
- ☐ **The agent acts on my paste the same session,** not "later."

Start smaller than this

Do not build all eight at once. Build one card with one toggle and one copy button, use it on real work tomorrow, and let the rest of the list grow from the moments it annoys you. That is how every piece of mine arrived. The annoyance tells you exactly what to build next, and it is never wrong.

You now have the whole loop: the routing rule, the seven parts, the format argument, and the prompt to build it. Nothing here needs my tools. It needs your agent and one afternoon.

WHO MADE THIS

About the human *at work.*

I am Boubacar. I spent a decade operating inside a Fortune 100, trained as a writer before I ever shipped a line of code, and today I run my own business on an AI system I built myself.

I wrote this because the review loop is the part of an agentic workflow nobody hands you, and it is the part that decides whether the agents actually save you time or just give you more to read. This is exactly how I work, every day, from my phone.

Build the system once. Then let it carry the boring part.

THIS IS FIELD GUIDE NO. 01. THE REVIEW LOOP IS ONE PIECE. THE REST OF THE SYSTEM, THE AGENTS, THE HANDOFFS, THE PROMPTS I RUN EVERY DAY, LIVES HERE.

humanatwork.ai/store