THE COMPANION PAPER

The Model Audit

A technical breakdown of one assistant's architecture

Companion to “I Was Mad At My Butler. I'd Given Him a Cheap Brain.”
michaelgaff.com/journal/the-cheap-brain

Architecture only. No keys, tokens, endpoints, or personal data appear in this document.

1. Overview and scope

This paper documents the architecture behind an always on Telegram assistant: a Mac mini running locally, connected to calendar, mail, reminders, and an internal file search tool, driven by a language model. It is the engineering record behind the essay “I Was Mad At My Butler,” written so another builder auditing a similar setup can check their own configuration against it.

Scope note: this is architecture, not credentials. No API keys, tokens, account identifiers, file paths outside the general shape, or any personal, donor, or patient data appear anywhere in this document. Where the essay names a person's private records, this paper only names the category of data and how it is handled.

2. Where the model was pinned

The assistant's language model was set to a small, low cost model built for fast classification and simple extraction rather than multi step reasoning. That choice was not visible day to day. It was buried in two separate places:

- A default value hard coded inside the bot's own source, chosen once during initial setup and never revisited.
- A duplicate value set as an environment variable in the script that launches the bot each morning, overriding the code default and reinforcing the same choice.

Two pins pointing at the same answer feel like redundancy. In practice they are two places a future fix has to remember to touch, and a single audit script that greps for the model name in exactly one of them will report a clean result while the other still holds the old value. The lesson for any assistant build: a model choice should live in exactly one place, read by everything else, so an upgrade is one edit instead of an easter egg hunt.

3. Memory design: why a variable is not a memory

The assistant kept conversational memory as a variable inside its own running process. That single design decision explains most of the symptoms reported in the essay: forgotten context, invented events, and a total reset of history after any restart.

- A variable lives only as long as the process that declared it. It is not written to disk, not backed up, and not recoverable after a crash or restart.
- The bot's own connection handling restarted the process automatically after a small number of connection errors, which is a reasonable resilience pattern for uptime and a quiet memory wipe for context.

- Because the failure mode looked identical to normal amnesia (“I don't recall you telling me that”) rather than an obvious crash, it read as a personality flaw in the assistant rather than a storage decision.

The fix is not exotic: persist memory to a file or a lightweight database with an append only log, load it back on boot, and treat the in memory copy purely as a cache. A restart should cost latency, not history.

4. Conversation caps

Three limits compounded the memory problem, each reasonable in isolation and punishing in combination.

- A hard cap of twenty conversational turns before the session was considered stale.
- A ceiling of six tool calling steps per task, after which the assistant gave up rather than continuing to work the problem.
- Single threaded execution, meaning one slow running task blocked the assistant from responding to anything else until it finished or timed out.

None of these caps are wrong on their own. Turn limits bound cost, step ceilings bound runaway loops, and single threading simplifies state handling. The failure was tuning all three for a budget model's short, cheap interactions and then never revisiting the numbers when the job description grew to include real reasoning and multi step tasks.

5. The nightly archive job and scheduled amnesia

Separately, the platform underneath the assistant had its own memory ceiling. When accumulated context grew past what the platform could hold, a nightly job archived the session and restarted the process fresh. This was the actual, intentional fix for a real problem: unbounded memory growth. But the fix for remembering too much was, in effect, a nightly decision to remember nothing. The job solved a capacity problem by deleting the thing that gave the assistant continuity.

A better version of the same job summarizes and compresses the older parts of a conversation into a persistent, structured note before trimming it, rather than discarding it outright. Capacity management and continuity are not actually in conflict; they were only in conflict in this particular implementation.

6. The fix: what changed

The replacement design, running on a dedicated always on machine, changes four things at once:

- Model tier: a frontier, reasoning capable model replaces the budget classification model, with the choice defined in exactly one place.
- Memory: state lives in files rather than in a running process, so a restart costs nothing and history survives crashes, reboots, and deploys.

- Shared knowledge: the assistant reads from the same notes, runbooks, and reference documents the operator already maintains, so it can look things up instead of guessing.
- Tools: real, general purpose tool access replaces a narrow, hand rolled loop with a fixed step ceiling, so longer tasks do not hit an artificial wall.

None of these four changes required new infrastructure. They required noticing that a small set of early, reasonable seeming defaults had quietly become permanent.

7. A short audit checklist

If you run an assistant of any kind, local or hosted, these are the questions worth answering out loud, not from memory:

- What model is actually configured, and is that value set in more than one place?
- Where does conversational memory live: a variable, a file, or a database? What happens to it on restart?
- What triggers a restart, and how often does that actually happen in practice?
- Are there turn, step, or time limits, and were they chosen for the workload you have today or the one you had when you first built it?
- Is there a scheduled job that manages memory or context size, and does it compress or does it simply delete?

A consistent complaint about an assistant is rarely a mystery. Almost always, it is a setting.