NERO PROJECT

Text the House - Always-On AI Aide

One message. Filed where it belongs.

Paul Schindler | Built with Claude | November 2026

Executive summary

I built an always-on AI agent that two household members reach over Telegram from anywhere. A text or voice note goes to the right list, reminder, calendar entry or task, and the agent confirms in one line. Every request type maps to a declared route in one configuration table, so a new capability means one new route, not a rewrite of the agent's instructions. The agent shares one memory with the house voice assistant, and when no one is at the keyboard, written rules and hard limits enforced in code fence what it may do. Each outage it has met since June 2026 produced a named fix, from a 3-minute watchdog to a backup model that runs on local hardware.

Live	30	3 min	15 min
since June 2026	declared routes in one table	watchdog health-check cycle	end-to-end reply test

What is in this report

Page	Contents
2	The problem, and the design choices I made and ruled out
3	How it works - the message flow, the architecture and the route table ("the switchboard")
4	Safeguards - what the agent may do alone and what always waits for a person
5	Reliability engineering - what broke, how I found it and what fixed it
6	Results, status and next steps, and what the project shows about how I work

Project at a glance

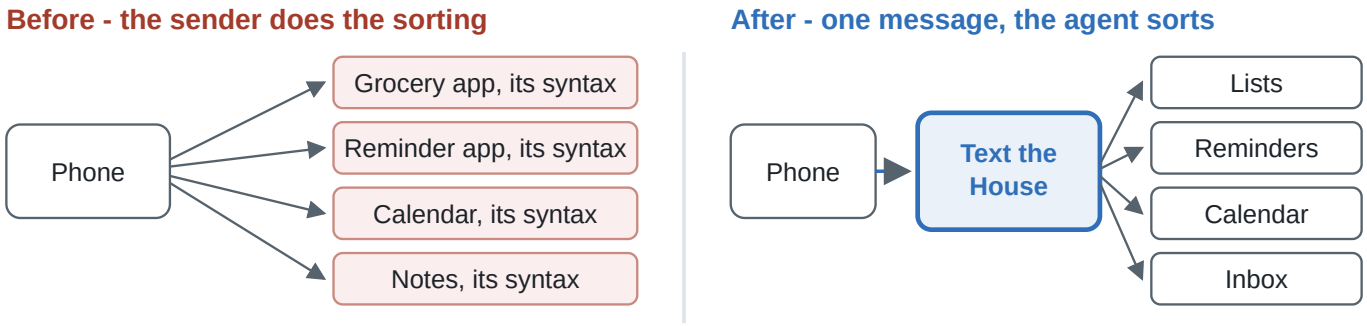
Who uses it	Two household members, each from their own phone
Ways in	Telegram text, Telegram voice notes and spoken requests to the house voice pucks
Timeline	Designed in my notes from May 2026. Live on Telegram since June 2026.
Where it runs	One always-on Claude Code session on a home workstation, restarted each morning
Where items land	Grocery and todo lists, reminders, the calendar and an inbox for anything else
My role	I set the requirements, designed the system, wrote its operating rules and verified every change by its outcome. AI agents (Claude) wrote the code.

1. The problem and why it mattered

A household runs on small commitments. Groceries, appointments, chores and ideas arrive at the wrong moment, usually away from a keyboard. Each one needs a home, and each home has its own app and its own syntax.

I wanted one place to send all of it. The rule I set from the start was zero memorized syntax. Nobody in the house should have to remember a command string, a list name or an app. A plain-English text or voice note had to be enough, and the agent had to say in one line where the item landed so the sender never wonders.

Two harder requirements sat behind that. First, an agent that runs unattended around the clock must have narrow, explicit authority, because no one watches it most of the time. Second, an always-on service is only useful if it is actually on. A channel that fails silently is worse than no channel, because people stop trusting it and go back to sticky notes.



The shift in one picture. The sender stops choosing an app and a format, and the agent states where each item went.

Design choices, and what I ruled out

Option considered	What I decided and why
Hard-coded command parser	Ruled out. Matching patterns cannot read intent. The AI model reads the route table and picks the best fit. The chosen route's one declared command is the only thing it runs, so the table, not improvisation, decides where data lands.
Routing rules written as prose in the agent's instructions	Replaced in July 2026 by one declared table with a validation check. A new capability is a new row, and prose instructions stay short.
Notes saved as loose text files	Ruled out. Captures land in structured local stores (lists, reminders, inbox, calendar) that other tools can read and that survive a restart.
One-time authenticator code to launch heavier jobs	Designed in June 2026, never built. In November 2026 I chose a cleaner split. The agent hands real work to a separate work session, and that session texts the requester when it starts and when it finishes.
Restart the agent when its working memory gets large	Ruled out in favor of one fresh start every day at 5:00 AM. A fixed daily restart is predictable, and no process can quietly accumulate days of conversation.
Give the agent its own settings file to isolate its permissions	Tested against the vendor documentation in July 2026 and rejected. Permission grants merge across settings files, so that design would have given false isolation. The correct design (a separate working folder) is recorded and deferred, because hard deny rules already block destructive commands.

The principle behind every choice

Durable memory lives in the stores and the written house rules, never in the conversation. That is what makes a daily restart safe. The agent files anything lasting the moment it arrives, so a restart drops only "what we just said".

2. How it works

1. Message
A text or voice note on Telegram, or a request spoken to a voice puck.

2. Match
The agent reads the route table and picks the route whose intent and examples fit best.

3. Run
It runs that route's one declared command. Nothing outside the table.

4. File
The item lands in a durable store - a list, a reminder, the calendar or the inbox.

5. Confirm
One plain line back, in the shape the route declares.



One session serves every channel. The voice pucks joined it in June 2026, so a spoken request and a text from the road reach one memory.

The switchboard - one table decides where everything goes

Each route declares an intent, sample phrases, one command and the exact confirmation. Four simplified rows show the idea.

Route	When it applies	Example message	Files to	Confirmation
`grocery`	A grocery or household item to buy later	"Add milk"	Grocery list	"Got it, on the grocery list"
`reminder`	A dated obligation or deadline	"Remind me: dentist Thursday 2 PM"	Reminders	"Reminder set for Thursday"
`calendar`	A real event with a time	"Dinner with friends Saturday 7 PM"	Calendar	"On the calendar"
`fallback`	Anything that matches no route above	A message that fits nowhere	Inbox	"Not sure where this goes - parked it in your inbox"

Adding a capability, in three steps

(1) Write one row. (2) Run the validation check on the whole table. (3) Feed the sample phrases to the matching test and confirm each lands on the new route. The agent's instructions point at the table, so they do not change. The catch-all row stays last, so no message is ever dropped.

3. Safeguards - narrow authority for an agent no one is watching

Most of the time no person sits at the keyboard. I treated that as the design constraint, not an afterthought. The agent can be reached by anyone in the household at any hour, so what it may do alone has to be short and written down, with hard limits enforced in code rather than left to the agent's judgment.

What it may do alone

- Add to the grocery and todo lists
- Set and read reminders
- Add events to the calendar
- Capture notes and ideas to the inbox
- Send a notice to a household member
- Answer questions and run read-only lookups

What always waits for a person

- Any change to a live system, which needs a keyboard session and a written change brief
- Real work, which goes to a separate work session that texts its start and finish
- Any delete or "done" whose target the agent had to infer
- Approving a new user of the bot
- Any high-value password, which never travels over chat

The rules, and the mechanism behind each

1. **Its unattended job list is written down.** House rules limit what it changes on its own to lists, reminders, the calendar, the inbox and notices. Anything else waits for a person.
2. **A route grants no new power.** A route can only name a command that is already allowed. Declaring a route in the table changes where data goes, never what the agent is permitted to run.
3. **Hard deny rules sit under everything.** A deny list and a guard script block destructive commands no matter what any permission file allows. Remote shell access and container launches are blocked outright for this agent.
4. **No destructive action on a guess.** When a message asks to delete an event or mark a reminder done and the agent had to infer which one, it repeats the full title and waits for a plain-word yes. This rule came from a July 2026 incident, described on page 5.
5. **Nothing is ever dropped.** A message that fits no route lands in the inbox with a one-line note saying so.
6. **Only approved accounts get in.** A stranger who finds the bot receives a pairing code and is ignored until someone approves it at the keyboard. A chat message cannot approve a pairing, which closes the obvious injection route.
7. **Untrusted outside text stays out.** Commands that would feed raw text written by strangers (for example, the body of merchant email) into the unattended agent are withheld. They run only where a person is present.
8. **Heavy work runs elsewhere.** Since November 2026 the agent hands real work to a separate work session through one launcher. That session runs in its own clean environment, survives a restart of the agent and texts the requester when it starts and when it finishes.
9. **Fewer messages, not more.** A burst of results becomes one message, never one per item. House-wide loud channels fire only on evidence the household has agreed is strong enough.

Why this matters beyond one house

The same question faces any team that lets an AI agent act on its own. The answer here is structural. Authority is a short written list, the platform enforces it, and every exception routes to a person who can see the full context.

4. Reliability engineering - each outage became a named fix

An always-on agent fails in ways a process monitor cannot see. The process can stay "active" while the agent is deaf. I added each layer of checking after an outage that the layer above it missed.

Check layer	Catches	Missed (next layer)
1. Is the process running? service manager, restarts on crash	a crash	messaging tool died inside a live process (June 2026)
2. Is the messaging link up? watchdog, every 3 minutes	a dead link	a dialog frozen on screen (July 2026)
3. Is the screen blocked? reads the screen each cycle	usage-limit screen, permission dialog	a hang that looks idle (July 2026)
4. Does it actually reply? test message every 15 minutes	anything that stops a real reply	the backstop

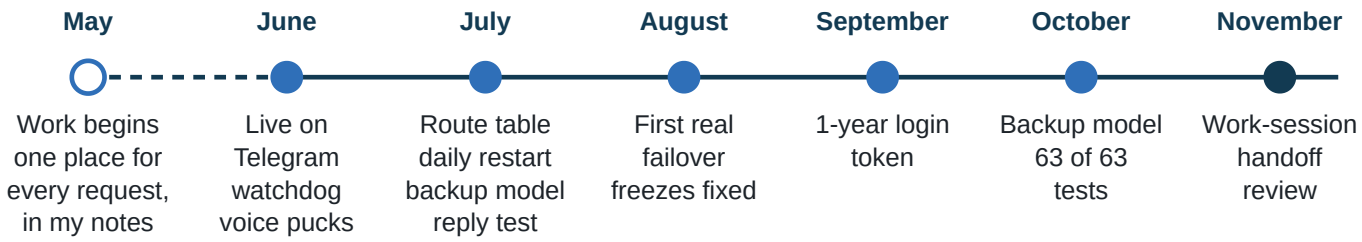
Each layer is narrower and closer to what a person experiences. The end-to-end test restarts the agent only after two misses 5 minutes apart, because one slow answer is normal. A broken test never restarts anything.

When	What broke	What fixed it
June 2026	The messaging tool died inside a running session. The service manager still said "active".	A watchdog every 3 minutes restarts the agent and alerts at most once an hour.
July 2026	The agent hit its cloud usage limit and sat on a dialog no one could press.	The watchdog detects that screen and hands the bot to a backup model running locally through Ollama. The backup says it is the backup and hands back after the limit resets.
July 2026	The agent hung for 22 minutes (4:45 PM to 5:07 PM) with every check green. Three texts went unanswered.	The end-to-end reply test. It saves the screen as evidence before any restart.
July 2026	"2 done", meant as an answer to a pending decision, marked a real reminder done instead.	A grammar rule in the route table, plus the no-guess rule for destructive actions.
August 2026	Seven freezes in one day on permission dialogs. A software update had changed the dialog, so the watchdog no longer recognized it.	The watchdog now recognizes a dialog by its structure, and a replay test runs every saved screen through it after each update.
September 2026	A shared login lapsed on a 30-day clock and logged out every unattended agent at once. The August lapse lasted 3 days.	A 1-year token, saved only by a script that proves it with a live call first, plus a renewal reminder.
November 2026	A work handoff launched the wrong way took over the bot's connection. The watchdog restarted it 6 times in 18 minutes.	One launcher for all handoffs, a guard that strips the bot's settings from any child process and a voice failure that can no longer take text down. Verified recovery took 2 minutes 20 seconds.

A restart triggered by the reply test first saves a capture of the agent's screen, and the watchdog keeps the last 10. Before that rule, evidence from a July 2026 hang was overwritten before anyone read it.

5. Results, status and what is next

Work began in May 2026, when I designed one place for every household request in my notes. The agent went live in June 2026 as a remote line into the home workstation and became a messenger the next day. The voice pucks joined the same session that month, the route table replaced prose routing in July and the work-session handoff went live in November.



All months are 2026. The dashed segment is design work in my notes. Each reliability step traces to a failure seen in real use. The October test ran 63 scripted cases after a code guard began building status replies from actual command output.

Item	Status
Messaging agent on Telegram	Live since June 2026, reachable by two household members
Shared memory with the voice pucks	Live since June 2026
Route table (the switchboard)	Live since July 2026, 30 routes, validated after every edit
Backup model on local hardware	Live since July 2026, first real failover in August 2026
Work-session handoff	Live since November 2026
Local clock line on every message and a "newest record wins" rule	Proposed November 2026, awaiting my decision. A review found the agent's recent slips came from reading an older record as the current one, with only Coordinated Universal Time (UTC) stamps in front of it.
Separate working folder for the agent's permissions	Designed in July 2026, deferred to a maintenance window

Three lessons I would carry into any AI deployment

1. Test the round trip, not the process. Every early outage passed the process checks.
2. Detection rules go stale. A software update silently retired one, so saved evidence now replays through the watchdog after each update.
3. Alarms must know about each other. One expired login once produced 18 error messages from five separate alarms before anyone named the cause.

6. What this shows about how I work

Text the House is a small program office. It has one intake, a published routing table, a defined fallback and a short written list of what may happen without sign-off. When it failed, I did not patch symptoms. I named the failure, found why the existing checks missed it and added the check that would have caught it. I bring the same discipline to partnerships and programs, with clear intake, explicit authority, honest status and a record of every fix. It is also a working model of AI adoption, where an agent earns autonomy one narrow, verified duty at a time.