

Grace: a memory that lets AI coding assistants learn from a correction once

Today's coding assistants forget everything when a session ends. Grace gives them a memory of their own work, so a mistake a reviewer corrects today is not made again tomorrow.

The problem

An AI coding assistant starts every session from zero. Correct it today ("on this team we never reuse an ID after a record is deleted") and tomorrow, in a new session, it can make exactly the same mistake. The team pays twice: another review round, and the work redone from scratch.

The usual fix, a notes file the assistant is told to read, works only when someone writes the right note and the model finds it.

What Grace is

Grace is a memory that runs on the developer's own machine, next to the assistant. Nothing goes to a cloud service.

- **It remembers the work as it happens:** each request, each step, and each time a reviewer rejects the work and says why.
- **It recognises work that comes back,** and hands the assistant what happened last time, including the correction, before any code is written.
- **It works with any model:** a model running on your own hardware (Google's Gemma) or a hosted one (Anthropic's Claude Code).

Inside, Grace is not a notes database or a search index: it is a small simulated network of neurons, modelled on the fruit fly's memory circuit, that learns as things happen and recalls by recognition.

How we test it. The same two-day test, run with Grace and without. Automated tests decide whether Day 2 was right the first time, and each study's rules are written down before it runs.

Day 1. The assistant builds a small notes app. The reviewer rejects it: "IDs must never be reused after a delete." The assistant fixes it, and Grace remembers the rejection.

Day 2. A brand-new session, in an empty folder, gets the same request. Before it writes code, Grace hands it the Day 1 rejection, and it gets the IDs right the first time.

Results: Google's Gemma 4 (26B), running locally on one machine

Tested on one AMD Ryzen AI 9 HX PRO 370 machine with integrated Radeon 890M graphics and no separate GPU (Ubuntu 26.04; Gemma 4 26B, 4-bit, through Ollama). Grace ran on the same machine.

STUDY	WITH GRACE: RIGHT FIRST TIME	ALONE: RIGHT FIRST TIME	TOKENS ON DAY 2 (WITH · ALONE)	TIME TO ACCEPTED (WITH · ALONE)
Oct 14	3 of 3	0 of 3	81K · 391K	6 min · 17 min
Oct 20, repeat	3 of 3	0 of 3	172K · 319K	10 min · 18 min
Oct 27, current release	2 of 2	0 of 3	50–122K · 377K–2.0M	4–7 min · 13–42 min

Working alone, Gemma repeated the corrected mistake every time; with a notes file (Oct 14) it was right the first time 1 of 3. Single figures are medians.

Results: Anthropic's Claude Code, four different tasks

Claude Code (Opus) runs in Anthropic's cloud; Grace ran on an Apple M5 Mac with 24 GB of memory. Each task had its own unstated team rule. In every run, Claude Code alone was also corrected on Day 1, kept its own settings, and was then given the same request on Day 2.

TASK	WITH GRACE: RIGHT FIRST TIME	ALONE, CORRECTED ON DAY 1 TOO: RIGHT FIRST TIME	COST PER DAY 2 SESSION, MEDIAN (WITH · ALONE)
Notes app	5 of 5	0 of 5	\$0.19 · \$0.48
To-do app	5 of 5	0 of 5	\$0.16 · \$0.35
Expense ledger	5 of 5	0 of 5	\$0.24 · \$0.57
Bookmarks	5 of 5	0 of 5	\$0.22 · \$0.45
All four	20 of 20	0 of 20	\$0.21 · \$0.47

Claude Code alone repeated the corrected mistake in all 20 runs; 20 of 20 against 0 of 20 gives $p = 7 \times 10^{-12}$ (Fisher's exact test, one-sided). Three of the four tasks were written independently, without access to Grace's design. With Grace, Claude used 45% fewer tokens on Day 2 (median 81K against 147K).

Why it matters

- **A local model that learns like a team member.** After one correction, Gemma with Grace, on one machine without a separate GPU, was right the first time in 8 of 8 runs, using a fraction of the tokens it needed alone.
- **About half the cost on repeat work:** the assistant stops paying for the same mistake twice.

A local memory, for any model, that lets an AI coding assistant learn from a correction once instead of repeating it.