A TWO-PAGE CHECKLIST

Give your AI a memory that survives every session

An AI assistant starts every session as a blank slate. This is the setup I run across all my projects so the work compounds instead of resetting. Four layers, six rules. The rules matter more than the tools.

The four layers

1. A short always-loaded index

One file the assistant reads at every start. It holds the always-apply rules and a one-line pointer to every memory. Hard size cap. Links out to detail, never the detail itself.

2. One fact per file

Small files, loaded only when relevant. Each has a name, a one-line description used for recall, and a type: user (who you are), feedback (a correction, with the why), project (state you cannot read off the code), reference (a pointer).

3. A searchable deep store

The long tail of decisions and small learnings. Queried on demand at the start of a task, never loaded by default. Any keyword or vector store works.

4. An automatic save

A hook that runs when a session ends and files what was decided, corrected and learned. Capture has to be automatic, or the note is never taken.

The nightly cleanup pass

A scheduled job reads which memories were used, then proposes merges and deletions. It never edits a memory on its own. You read the proposal and approve it. A fact nobody has read in months is a candidate to go.

A memory file, as an example

```
---
name: invoice-number-format
description: Invoice numbers use the prefix INV-2026- and a number is never reused
type: feedback
---
Owner corrected this on 2026-03-14 after a duplicate number went out.

Why: the accountant reconciles by invoice number, so a reuse breaks the match.
How to apply: when generating an invoice, read the last number and add one.
```

SIX RULES

What keeps the memory from rotting

- [ ] **Do not store what the code or the files already record.** Structure, past fixes and history are readable on demand. Memory is for the non-obvious: the why behind a decision, a preference, a constraint that lives in someone's head.
- [ ] **Write dates in full.** "Last week" is wrong the moment it is saved. "2026-05-20" stays true.
- [ ] **Update, never duplicate.** Before saving, look for an existing file on the same topic and edit that one. Two memories that disagree are worse than none.
- [ ] **Treat a recalled memory as context, not gospel.** A note that names a file, a flag or a price may describe something that has changed. Verify it still exists before acting on it.
- [ ] **Separate always-apply rules from situational ones.** Posture and safety rules stay in the always-loaded file. Tool-specific lessons load only when that kind of work is happening.
- [ ] **Prune.** Delete memories that turn out to be wrong. A memory system you never garden becomes one you stop trusting.

Set it up this week

- [ ] Write the always-loaded file. Keep it under a hard line cap. Rules first, then the index.
- [ ] Create the four folders or tags: user, feedback, project, reference.
- [ ] Move three things you keep re-explaining into three one-fact files.
- [ ] Add the end-of-session save. If your tool has hooks, use one. If not, a template you fill at the end of each session is the fallback.
- [ ] Put the cleanup pass on a schedule. Propose only. You approve.
- [ ] Thirty days later: delete every memory you did not use.

The honest limits

Memories go stale, and a setup with no pruning fills with facts that were true once. Treated as a living system with occasional gardening, this is one of the highest-leverage things you can build around an AI assistant. Treated as a write-only archive, it becomes a liability you stop reading.

Want this wired into your own business?

I build AI systems for companies, with memory that compounds instead of resetting. The full write-up is at **cweihan.com/notes/claude-code-memory-setup**. For a business inquiry, message or WhatsApp **019-368 8489**, or send a brief at **cweihan.com**.