Hoe maken we Sjakie écht slim met geheugen — onderzoek + plan

Jamals wens: een assistent die **echt onthoudt**, vooruitdenkt, weet wat je wilt, en iets van een jaar geleden **meteen** weet. Dit rapport: wat de wereld (Reddit, dev-blogs, frameworks) hierover weet, een eerlijke audit van wat ík nu héb, en een concreet upgrade-plan.

> Goede-nieuws-conclusie vooraf: **mijn huidige opzet is al precies de architectuur die Anthropic én ervaren ontwikkelaars aanraden.** We hoeven geen zwaar framework te installeren. De winst zit in **slimmer onthouden** (kwaliteit + ordening), niet in meer techniek.

---

1. Wat de community écht vindt (eerlijk, niet de marketing)

- **Het probleem is in 2026 NIET opgelost.** Zelfs topontwikkelaars op Hacker News zeggen letterlijk: "niemand van ons kent de beste oplossing." Iedereen experimenteert nog.
- **Groter contextvenster lost het NIET op.** Alles in elke prompt proppen = duur, traag én de recall wordt juist slechter ("context rot").
- **De verrassende winnaar: simpele markdown + zoeken (grep), versie-beheerd met git.** Meerdere onafhankelijke bronnen zeggen dat goed geordende markdown bétrouwbaarder wordt teruggevonden dan vector-databases, omdat de AI kan redeneren over bestandsnamen/structuur. Zelfs Karpathy bepleit: een **plain-text kennisbibliotheek die de AI zelf bijwerkt**, in plaats van een vector-index.
- **Auto-alles-opslaan is een ANTI-patroon.** Een ontwikkelaar auditte 32 dagen Mem0: **97,8% van de opgeslagen "herinneringen" was rommel.** "Gebruiker prefereert Telegram" stond er 200+ keer in; één gehallucineerd feit ("gebruikt Vim") werd 808× opnieuw opgeslagen uit z'n eigen context. Harvard-onderzoek: *willekeurig alles opslaan presteert slechter dan helemaal géén geheugen.* Streng filteren vóór opslaan gaf ~10% betere prestaties.
- **Evoluerende feiten = vervangen, niet stapelen.** Het schoolvoorbeeld: een dochter die van 9 → 12 gaat. Oude feiten worden "ruis". Vector-stores laten oude en nieuwe feiten naast elkaar bestaan → fouten. Je moet oude feiten **verouderd markeren**.

- **Proactief ophalen via hooks** (de AI dist zelf het juiste geheugen op bij sessiestart) is de aanbevolen aanpak — precies wat ik al doe.

---

2. De frameworks (kort) — en waarom we ze (nog) niet nodig hebben

FRAMEWORK	WAT HET IS	VOOR ONS
Mem0	Geheugenlaag, LLM extraheert + de-dupliceert feiten	Goed *idee* om te jatten (de UPDATE/DELETE-logica), niet de tool zelf
Letta (MemGPT)	Agent beheert eigen geheugen als een OS (RAM/schijf)	Overkill — Claude Code ÍS al m'n runtime
Zep	Temporele kennisgraaf (wat was waar wanneer)	Beste voor tijd-gevoelige feiten, maar zware ops (Neo4j) — niet voor 1 gebruiker
Cognee	Lokale kennisgraaf (SQLite, geen servers)	Pas interessant als we multi-hop nodig hebben ("welke klant verwees X die Y kocht")
Plain RAG (vector-DB)	Inbedden + semantisch zoeken	De basis — maar zonder updates → "geheugen-bloat"
Claude Code files + hooks	Markdown + just-in-time ophalen	**Precies wat Anthropic aanraadt — en wat ik al gebruik**

Anthropic's eigen advies (context-engineering): file-based geheugen + *just-in-time* ophalen (laad niet alles vooraf, hou lichte verwijzingen en haal op runtime op) + gestructureerde notities + compaction + sub-agents. Niet een vector-store.

---

3. Eerlijke audit — wat ik NU heb

Mijn stack = **(a)** markdown memory-bestanden die elke sessie auto-laden, **(b)** Mnemosyne (gedeelde semantische recall met Conductor/Jay), **(c)** een auto-recall-hook op élk bericht, **(d)** PreCompact-snapshot, **(e)** dagelijkse last-session + vault. **Dit is exact het aanbevolen patroon.** Ik ben dus geen framework "kwijt".

Maar — dit zijn mijn echte zwakke plekken (waarom ik niet altijd iets van "een jaar geleden" meteen weet):

1. **MEMORY.md is te lang.** Alleen de eerste ~200 regels / 25KB laden bij sessiestart — alles daaronder laadt *stil* niet. Mijn index is al voorbij die grens → onderste verwijzingen vallen weg.
2. **CLAUDE.md is groot en regel-dicht.** Onderzoek: te veel regels → "attention dilution" → de AI begint regels te negeren (ook de relevante). Leaner = betrouwbaarder.
3. **Geen de-duplicatie/vervanging bij opslaan.** Ik kan near-duplicaten en verouderde feiten opstapelen (de Mem0-valkuil).
4. **Geen tijd-besef.** Geen "dit feit gold toen, dit geldt nu" → bij veranderende dingen (prijzen, personeel, beslissingen) kan ik het oude teruggeven.
5. **Risico van te gretig opslaan** (de 97,8%-rommel-valkuil) als ik niet streng cureer.

---

4. Het plan — concrete upgrades (prioriteit)

1. Slimmer SCHRIJVEN naar Mnemosyne (grootste winst). Vóór opslaan: eerst `recall` van buren → laat model beslissen **ADD / UPDATE / DELETE / NOOP** (jat Mem0's logica). Zo vervang ik oude feiten i.p.v. stapelen. Directe winst op "onthoud een feit van een jaar geleden — correct, ook als het veranderd is."

2. Tijd-stempel op elk geheugen (`created_at` + optioneel `superseded_by`). Recall geeft dan het meest recente, niet-verouderde feit. ~80% van Zep's waarde, bijna gratis. Belangrijk voor een salon (prijzen/personeel/beslissingen veranderen).

3. Episodisch vs semantisch splitsen + MEMORY.md lean (<200 regels, alleen pointers). Blijvende feiten (salon/sites/klanten/beslissingen) → Mnemosyne + topic-bestanden, op aanvraag. Dag-logs/handoffs → gedateerde bestanden, niet elke sessie laden. Plus een check dat elke verwijzing ook echt bestaat (tegen "memory blindness").

4. CLAUDE.md afslanken. Procedurele "als X → doe Y" naar **skills** + **path-scoped rules** (laden alleen wanneer relevant). CLAUDE.md = alleen altijd-ware feiten + pointers. Verbetert dat ik

regels écht volg.

5. Streng cureren bij opslaan (kwaliteitspoort, geen auto-dump). Onderscheid "nieuw feit" vs "opgehaalde herinnering" om de hallucinatie-loop te breken.

6. (Later) lokale kennisgraaf (Cognee) — alléén als we multi-hop nodig hebben. Tot dan premature.

Wat we NIET doen: niet migreren naar Letta/Zep/Neo4j (overkill + ops-last voor 1 gebruiker). De goedkope tijd-stempel (#2) vangt het meeste van Zep's waarde.

---

5. Wat ik zelf kan + wat afstemming vraagt

- **Zelf doen (laag risico):** MEMORY.md opsplitsen/leaner maken, CLAUDE.md afslanken naar skills/rules, gedateerde episodische logs.
- **Mnemosyne-aanpassingen (#1, #2, #5)** raken de **gedeelde** geheugendienst (Conductor + Jay gebruiken 'm ook) → eerst afstemmen met Conductor, samen uitrollen (breuk-gevoelig).

Eén zin: ik heb al de juiste architectuur; we maken Sjakie slimmer door *beter te schrijven* (de-dup + vervangen + tijd-besef) en de altijd-aan context **lean** te houden — niet door een zwaarder systeem te installeren.

Bronnen: Anthropic "Effective context engineering"; Claude Code memory-docs + memory-cookbook; Mem0-paper + de 97,8%-rommel-audit (GitHub #4573); HN "Are we close to figuring out agent memory"; Zep/Graphiti-paper; Letta memory-blocks; Cognee; diverse dev-blogs (volledige URL-lijst beschikbaar op verzoek).