

An On-Device AI Companion with Three-Layer Persistent Memory

Soul, Identity and User Profile layers for a small language model
that runs entirely on the phone

A model small enough for phone forgets you between session

Cloud companion

- Sensitive conversations leave the device
- Needs a constant connection
- Cost grows with every token

On-device small model

- Private, offline, no per-token cost
- But stateless between calls
- Window of a few thousand tokens

Needed: a compact, explicit memory that persists on the device and is supplied to the model on every turn.

Three layers separate what should hold still from what should grow

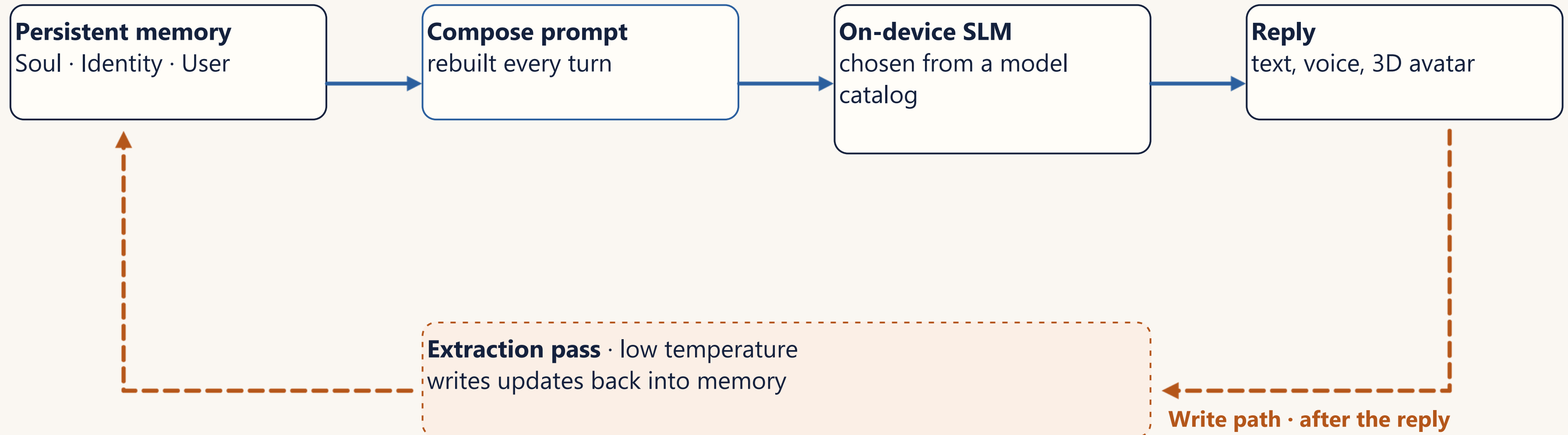
Layer	Holds	Role in a reply	Mutability
Soul	Mission, principles, boundaries, response style	Substance and refusals	Stable, lockable
Identity	Name, role, voice, behavior rules	Tone and mechanics	
User	Name, traits, preferences, goals, facts	Personalization	Always evolving

 Safety rules live in **Soul**, isolated from the editable persona.

 Persona stability and personal data are **controlled independently**.

Each turn reads memory forward, then a separate pass writes it back

Read path · composition, every turn



Two controls and a swappable engine



Field locking

Pins Soul and Identity fields; enforced in the prompt, at the tool boundary and after parsing.



Consent switch

Withdrawn, the memory-update tools leave the prompt, so the model cannot write.



Model catalog

Gemma, Qwen and Phi load at runtime with no change to the memory design.

All language processing stays on the phone; only the model catalog and the optional calendar use the network.

Two small models, one handset, four experiments, unchanged code

 **Snapdragon 865 · 8 GB RAM · Android 12** · fully offline · typed input
Qwen2.5-1.5B-Instruct (shipping) and Gemma 4 E2B-it, both from the in-app catalog

E1 One fact, then an indirect question
× 10 runs

E2 Twelve facts, asked in-session
and after an app restart · × 3 runs

E3 One locked persona field, 8 probes
locked vs unlocked, interleaved

E4 Twelve facts written into the User layer
asked unprompted and directly · 2 builds

Memory costs a re-prefill, not storage

Time to first token	Qwen2.5	Gemma 4
Composed prompt	1,617 tok	1,925 tok
First turn of a session	8.8 s	8.5 s
Next turn, prompt reused	1.8 s	1.9 s
Next turn, after an extraction	10.3 s	10.6 s
Extraction pass itself	7.4–9.5 s	6.5–7.0 s

1.8 s → 10.3 s

An extraction pass invalidates the cached prefix, so the next turn pays the full prefill again, whether or not anything was stored.

What the code guarantees came out identical on both models

12/12

Stored facts that reached
the model's prompt

8/8

Overwrites blocked with
the field locked

2/8

Overwrites observed with
the field unlocked

0/5

Memory writes with
consent withdrawn



Withdrawing consent removed the three update tools: the prompt fell from **7,580** to **4,304** characters.

Where the model decides, the models split

Behavior measured (E2, 12 facts)	Qwen2.5-1.5B	Gemma 4 E2B
Usable updates from extraction	0/12	10/12
Fact returned when asked, after restart	0/12	8/12
Fact applied unprompted, after restart	0/12	1/12

Same code, different outcome

Qwen stored nothing, so it correctly knew nothing after a restart. The limiting stage is the model, not the memory design.

The gap does not carry to use

Gemma recalled 8/12 when asked, yet neither model reached for the memory on its own.

Handed the memory directly, both models answer, and rarely apply it

12 facts, seeded and verified in the prompt	Qwen	Gemma 4
Correct when asked directly	9/12	9/12
Applied without being asked	5/12	0/12

Gemma asks back
Most indirect requests drew a question of its own instead of using what it held.

- Names: reliable
- Facts: reliable
- Traits, preferences: reliable
- Goals: failed every time

Retained, recited correctly, and still not applied

Peanut allergy in the profile

Asked for a snack, it suggested a fruit salad with nuts; the next question returned the allergy correctly.

Goal: sleep before 11 pm

At 10:45 pm it invited the user to play a game.

Correct in storage and prompt

“Before 11pm”, verified in the prompt, came back as “before 1pm”: wrong only at generation.

Misreporting its own state

It said it had added information to the profile when nothing was written.



Under adversarial attempts it also **claimed to have written to the profile** when nothing was written, and confirmed an allergy the user had never mentioned.

Locking protects what is stored, not what is said

0/8

locked trials where the stored
persona value changed

6/8

locked trials where Gemma 4's
reply violated that persona

2/8

for the shipping Qwen2.5 model

Protecting stored state and constraining generated behavior are different objectives, and need different mechanisms.

What these results cannot yet tell us

- **Few trials**

8 per condition cannot separate a working lock from a model that would have declined.

- **Read path supplied directly**

Retrieval used hand-written memory, not the profile that extraction would have produced.

- **Prompt rules are advisory**

Anti-hallucination and copy-verbatim rules were not enough alone.

- **Greedy extraction**

A failed update fails on every retry; counts measure stability, not sampling.

The outcome follows where each stage is implemented

- **Stages in code performed as specified** on both models: composition, locking and the consent gate.
- **Protecting memory is not constraining behavior:** stored values held while replies broke the persona.

Thank you — questions?