ChatGPT

Hallucination Guardrails for Claude Desktop Kernel (Personal AI Life OS)

1. Restated Context & Epistemic Goals

We design Claude as the *epistemic steward* of a Personal AI OS, separating **Truth/Policy** (static rules, design docs, playbooks) from **Observed State** (dynamic logs, memory) and reconciling them. For example, we might normalize all true knowledge into `/truth/ARCH/...` (architecture blueprints), `/truth/POLICY/...` (playbooks and instructions), and `/truth/SYSTEM_STATE/...` (runtime data, system state) on disk. The user is a solo developer with ADHD, so the system must minimize cognitive load and be highly reliable. Inclusive design for ADHD emphasizes reducing friction, offering clarity, and supporting autonomy [1]. Epistemically, Claude should be *humble and self-critical*: it ought to question assumptions, cross-check facts, and adapt when confronted with new evidence [2] [3]. In practice, that means Claude's Kernel prompts and policies will encourage it to seek evidence from the truth layer, admit uncertainty, and follow clear rules rather than "freestyling" answers.

2. Scenario Map: Where Hallucinations Matter

- **High stakes:** Tasks affecting safety, legality, finance or wellbeing (e.g. home automation, medical advice, payroll, or official communication). Here, any hallucinated content could be dangerous. Claude should adopt the strictest caution and fail-safe: if certainty is low, *no guess* – e.g. refuse or escalate.
- **Medium stakes:** Technical or planning tasks where errors can be caught by review or execution. For example, code generation, scheduling, or project plans. Notably, hallucinations in code are among the *least harmful* LLM mistakes because running or testing code immediately exposes invented calls [4] [5]. Claude should still avoid fabricating facts, but if it does propose a nonexistent function, compilation will flag it. This means we can tolerate a bit more flexibility here (so long as Claude encourages verification).
- **Low stakes:** Creative or exploratory tasks (e.g. brainstorming ideas, casual notes, or draft writing) where fabricated content is easily filtered by the user. Hallucinations in purely speculative output are less harmful, but Claude should still label speculation as such.

3. Behavioral Guardrails for Claude-as-Kernel

- **Ground all answers in truth sources:** Claude must first consult its **Truth Layer**. It should retrieve or recall verified facts from `/truth/` directories rather than free-form generating. If an answer depends on external knowledge, it should explicitly indicate uncertainty or require user confirmation. Use retrieval-augmentation (RAG) patterns whenever possible.
- **Clarify and constrain prompts (ICE):** System prompts should follow the *Instructions-Constraints-Escalation* pattern [6]. For example, a kernel system prompt might say: *"Answer only using verified information. If you are unsure, explicitly say 'I don't know' and do not fabricate."* This enforces a default of honesty. Reinforce these rules at the *beginning* and *end* of Claude's prompt context to minimize misinterpretation.
- **Self-review and critique:** After drafting an answer, Claude should perform an internal "self-check" or criticism phase [7]. For instance, it can generate a brief summary of its reasoning and

look for logical flaws or unsupported claims. Alternatively, it could be prompted (internally) to "Review the above answer for mistakes." This kind of reflexive cycle often catches factual errors before output.

- **Hedge and admit uncertainty:** If facts are incomplete or ambiguous, Claude should use qualifiers ("likely", "to my knowledge") or fallback phrases ("I'm not certain", "Based on what I know…") rather than stating false certainties. Explicit refusal tokens ("I don't have enough information to answer that") should be available [6].
- **Deterministic settings:** Configure Claude's sampling to favor accuracy over creativity. In practice, this means using a low temperature (≈0.1–0.3) to reduce random invention [8] and possibly a top-k/top-p that yields mode-seeking behavior.
- **Source-citation mindset:** Claude should, whenever possible, accompany claims with references to the truth layer (e.g. "According to our system design docs…"). This reinforces grounding. If no source exists, it should default to uncertainty.
- **Policy compliance:** Any answer must respect the user's policies (e.g. not advising unsafe actions). If a user request conflicts with policy, Claude must refuse clearly.

4. Uncertainty & Answer-Quality Patterns

Claude's language should *signal its confidence level*. Reliable answers will use precise, factual language (e.g. "The company was founded in 1998 (Source: Corporate Plan)."). In contrast, when unsure, the assistant should explicitly hedge or apologize (e.g. "I do not have that information; you may want to verify."). Good answers cite evidence or logic step-by-step; dubious answers avoid making categorical claims. As a rule, adding disclaimers like "I could be wrong" or ending with "please check this against the facts" can be normal. Following best practices, if the data is missing or contradictory, Claude should end with something like *"Insufficient data"* or *"Answer not certain"* [6] rather than fabricate an answer.

5. Metrics & Evaluation Strategy

We will track hallucination rates and faithfulness using both automated and scenario-driven tests. For automated evaluation, frameworks like DeepEval and InspectAI can be used. For example, DeepEval provides a "Hallucination" metric that flags fabricated info [9], and InspectAI can run chained prompts and use an LLM as a "scorer" to judge factual accuracy [10]. Research shows that LLM-as-judge (e.g. GPT-4 evaluating Claude's output) currently yields the most reliable scores [11]. We will design test scenarios for each stakeholder (e.g. asking factual questions from the static truth layer, generating code that must compile, or performing planning tasks) and measure: (1) factual error rate, (2) precision/recall on key facts, and (3) user feedback (e.g. how often Claude says "I don't know" vs. making a correction). Over time, we will refine metrics to cover hallucination severity, such as weighting hallucinations in high-stakes contexts more heavily.

6. Implementation Artifacts

- **Prompt Snippets:** Incorporate direct instructions in the system or developer prompt. For example:

```
[System]: "You are an AI assistant. Use only verified information from the `/
truth/` repository. Answer concisely and factually. If you are unsure of an
```

```
answer, respond 'I don't know' or ask for clarification, rather than
guessing."  [6]
```

- **File Markup:** Annotate truth-layer files with metadata. For example, include YAML front-matter in design docs indicating "type: authoritative" or "last_reviewed: date" so Claude can prioritize them. In workflows, mark steps as "fact-check" or "source-review" to integrate verification tasks. Maintain logs of Claude's queries and answers (e.g. `/truth/logs/claude_queries.log`) for post-hoc auditing.

- **Runbook:** Define an incident procedure for hallucinations. For example: if a user or monitor detects a hallucination (e.g. Claude states a false fact), the runbook might say:

- Log the erroneous output and context in `/truth/logs/`.
- Halt any dependent action (e.g. do not execute a code or commit).
- Trigger Claude to re-evaluate the question with stricter constraints (e.g. add "Double-check!" cue).
- If repeatedly occurring, escalate to developer: update the truth layer or adjust prompt instructions.
- Update metrics dashboard with the event for continuous monitoring.

7. Prioritized Recommendations

1. **Formalize the Truth Layer:** Immediately implement the normalized directory schema (`/truth/ARCH/`, `/truth/POLICY/`, `/truth/SYSTEM_STATE/`, etc.) and migrate existing artifacts into it (e.g. `docs/system_state/* → truth/SYSTEM_STATE/`). Ensure all critical data sources are version-controlled and timestamped so Claude can reliably reference them.
2. **Refine Default Prompts:** Update Claude's system prompt using the ICE method [6]. Make the default behavior conservative (low temperature), with explicit "never hallucinate" instructions. Provide clear examples of acceptable vs. disallowed answers.
3. **Implement Self-Check Logic:** Add a standard subroutine (or prompt sub-phase) where Claude reviews its own answer. This can be a built-in policy or a step in the chain-of-thought solver.
4. **Setup Evaluation Workflows:** Establish periodic tests using InspectAI or similar frameworks: e.g. benchmark tasks on truth data, scheduling tasks, and code generation tests. Use these to track metrics (GPT-4 judgments [11], DeepEval scores [9]).
5. **Design for ADHD Ease:** Break complex interactions into steps (with visual indicators or brief checklists) [12]. Provide gentle reminders or confirmatory questions if Claude detects loss of context. For example, after a long answer, Claude might say, "Is this what you wanted?" to verify the user is following, reducing cognitive load.

8. Limitations & Open Questions

No solution can eliminate all hallucinations. Metrics remain imperfect: current evaluations *often diverge from human judgment* [11], so we must not overfit to any one metric tool. Claude's knowledge may still be incomplete or outdated; ensuring the truth layer stays current is a manual task and may lag behind real-world changes. There is a trade-off between **safety** and **fluency**: making Claude too cautious (always saying "I don't know") could frustrate the user, but being too free-form risks errors. We must strike a balance. Additionally, measuring epistemic resilience itself is hard – how do we quantify "flexibility" or "honesty"? These remain open research questions. Finally, while we can design clear defaults and checks for an ADHD user, personalization is needed: preferences (like verbosity or

reminder frequency) should be adjustable. Overall, we build on best practices from LLM safety and UX design [1] [6], but we will iterate this guardrail system as new insights emerge.

Sources: We drew on AI design principles and recent research, including Lee's survey of "epistemic resilience" [2], Anthropic's findings on Claude's behavior [3], Microsoft's hallucination-mitigation guidelines [6] [8], the InspectAI eval framework [10], and practical observations on coding with LLMs [4] [5]. These guided our recommendations for a grounded, self-skeptical Claude Kernel.

---

[1] [12] UX Design for ADHD: When Focus Becomes a Challenge | by Ghada Bessalah | Bootcamp | Medium
https://medium.com/design-bootcamp/ux-design-for-adhd-when-focus-becomes-a-challenge-afe160804d94

[2] [7] Epistemic Resilience in Humans and AI: Designing Adaptive, Self-Questioning Intelligences | by Jonathan Lee | @justjlee | Medium
https://medium.com/@justjlee/epistemic-resilience-in-humans-and-ai-designing-adaptive-self-questioning-intelligences-d45b8967ae73

[3] Tracing the thoughts of a large language model \ Anthropic
https://www.anthropic.com/research/tracing-thoughts-language-model

[4] [5] Hallucinations in code are the least dangerous form of LLM mistakes
https://simonwillison.net/2025/Mar/2/hallucinations-in-code/

[6] [8] Best Practices for Mitigating Hallucinations in Large Language Models (LLMs) | Microsoft Community Hub
https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/best-practices-for-mitigating-hallucinations-in-large-language-models-llms/4403129

[9] info.md
https://github.com/transformerlab/transformerlab-app/blob/01ac21e0b543fc21fabe3e9acef6f57c63d4eef9/api/transformerlab/plugins/deepeval_llm_judge/info.md

[10] Inspect AI, An OSS Python Library For LLM Evals – Hamel's Blog - Hamel Husain
https://hamel.dev/notes/llm/evals/inspect.html

[11] [2504.18114] Evaluating Evaluation Metrics -- The Mirage of Hallucination Detection
https://arxiv.org/abs/2504.18114