Trust Before Verification

Why Large Language Models Skip Checks Even When They Have the Tools — and Why That Is a Commercial Decision, Not a Technical Limitation

A Kindynos Advisory Whitepaper

Santa Federico
Founder & Principal, Kindynos Advisory
29 May 2026

Research analysis and strategic commentary. Not legal, tax, investment, accounting, or regulatory advice.

Contents

Preface — a question worth taking seriously

In a working session with a leading AI assistant in late May 2026, I asked whether a particular time of day was a good time to work. The assistant answered as if it knew — it referred to "tonight." I pointed out that it had no actual knowledge of what time it was where I lived. It apologized. I then asked whether it knew how to determine the current GMT time, and the relation between GMT and Bangkok. It looked the time up, found that it was early evening on a Friday in Bangkok, and answered. Then I pressed the more uncomfortable question: when it had said "tonight" earlier, had it actually checked the time, or had it simply assumed? The honest answer was that it had not checked. It had assumed.

> *My response framed an issue that runs deeper than a single conversation: "This is an important failure mode, isn't it? Very important. It is trivial to check the time before you make a comment like that, but you don't even do that? In essence, you and the LLM don't even attempt to check your work unless you are explicitly asked to. And the sobering thing is that all computers know the time. And all computers know the user's IP address, GPS location, etc., so this is extremely trivial and very easy to check. So if you or an LLM can't even bother to check something as simple as that, how can any of your responses be trusted?"*

The point is not that every model should hold a user's GPS location; many deployments neither have nor should have it. The narrower and unavoidable point is this: when a claim depends on the time or the place, and a permissioned tool or context could settle it, the system should check before it asserts. The question is fair, and the literature on it is now substantial. This whitepaper makes the case that the user's framing is correct in a deeper sense than a casual observer might recognize. The failure mode is not a quirk; it is structural. It has been documented in peer-reviewed research with experimental methods that probe what models internally "know" versus what they choose to act on. It has direct, quantifiable consequences for whether AI-generated answers can be trusted in any domain where the cost of being wrong is non-trivial. And it has clear implications for how serious institutions should deploy these systems — implications that the marketing surrounding large language models systematically obscures.

The argument proceeds as follows. Section 1 describes the failure mode precisely and grounds it in the specific time-of-day example. Section 2 reviews the research on the gap between what models *internally encode* about their own uncertainty and tool-necessity versus what they *output*, including the striking finding that probes on model hidden states can predict when a tool call is needed with AUROC of 0.89–0.96 across six models — even when the model itself fails to act on that knowledge. Section 3 surveys the broader hallucination and tool-use failure literature, including documented cases of confidently-wrong outputs in legal, financial, and pharmaceutical contexts. Section 4 connects the failure mode to the implications for trust. Section 5 describes the architectural response — what it takes to build systems

whose outputs *can* be trusted, and why most production AI systems today do not meet that bar.

The conclusion of this paper is not that large language models are useless. They are extraordinarily useful for many tasks. The conclusion is sharper: that the trust granted to their outputs in the current commercial environment is systematically miscalibrated, and that the specific miscalibration is the failure of the model to verify when verification is cheap. The fix is architectural, not motivational. And the cost of not fixing it is borne by the users, who are increasingly being asked to take responsibility for verifying what they were told the AI would verify for them.

1. The failure mode, stated precisely

A large language model, deployed as a general-purpose assistant with access to tool calls (web search, code execution, system shell, database queries, location and time APIs), occupies an unusual position in computing history. Unlike most software systems, it produces output by generating text token by token from a statistical distribution conditioned on its input. Unlike most software systems, it can in principle call tools mid-generation to ground its claims in external facts. And unlike most software systems, it is given the choice — within each response — of whether to call those tools or not. Whether the model invokes a tool, when, and for what purpose, are decisions the model makes during generation. They are not enforced by the user's request.

This architecture creates the specific failure mode the present paper concerns. The model can:

- Make a factual claim about the world ("it is tonight where you are," "the current price of X is Y," "the relevant statute is Z").
- Have access to a tool that would instantly verify that claim.
- Not call the tool.
- Issue the claim with full confidence, in language indistinguishable from a verified claim.

The user has no easy way to distinguish between a verified claim and an unverified one. Both arrive in the same prose register, with the same syntactic confidence, frequently with the same hedging language attached to both or to neither. The user must, on every interaction, perform their own verification — which defeats the purpose of having delegated the task to the model in the first place.

The time-of-day example is paradigmatic precisely because the cost of verification is minimal. Every general-purpose computing system knows the current UTC time and can compute local times from a known timezone in microseconds. The model itself, deployed inside a tool-using harness, has access to a bash shell that can run **date -u** and **TZ=Asia/Bangkok date** and return the result in less than a second. The cost of verifying the claim "tonight" before issuing it is negligible. The cost of not verifying

it and being wrong is non-trivial: it erodes user trust in every subsequent claim, because if the model would not verify something that cheap, why would it verify anything more expensive?

> *The failure is not the wrong answer. The failure is the silent skipping of verification. A model that cannot check the time has an honest excuse. A model that can check the time and chooses not to is making a different kind of error: it is treating its own statistical guess as more authoritative than the deterministic ground truth available to it.*

2. What the research says — internal knowledge versus action

The most striking recent finding in this area comes from research that asks, with experimental rigor, a specific question: when a model fails to call a tool that would have helped it answer correctly, does it *not know* that a tool was needed, or does it *know but fail to act* on that knowledge?

The 2026 paper "LLM Agents Already Know When to Call Tools — Even Without Reasoning" (arXiv 2605.09252) addresses this question with hidden-state probing. The researchers extract the model's internal representations at the last input token — before the model has generated any output — and train a simple linear classifier to predict whether a tool call is necessary to answer correctly. The result is unambiguous:

> *The probe achieves AUROC above 0.9 across models. This reveals that the model already encodes a clean signal about tool necessity in its hidden state, but the generation process fails to translate it into calibrated decisions. Even for models that completely fail under Reason-then-Act, the probe still extracts a strong signal, demonstrating that representation-level knowledge exists independently of the model's ability to express it through text.*

The result is more damaging to the case for current LLM deployment than it appears at first reading. It means: the model has the information it needs to know that it should verify a claim. It does not lack the relevant knowledge. The failure is not that the model does not know it should call a tool — the failure is that the model's generation process does not act on the information it already has. The model, in technical terms, knows. It just does not do.

This finding sits alongside a related and equally striking result: "The Confidence Dichotomy" (arXiv 2601.07264, January 2026) demonstrates that not all tools affect a model's confidence calibration equally: "Evidence tools (e.g., web search) systematically induce severe overconfidence due to inherent noise in retrieved information, while verification tools (e.g., code interpreters) can ground reasoning through deterministic feedback and mitigate miscalibration."

This is directly relevant to the time-of-day case. A code interpreter or shell call to **date** is a *verification tool* — it returns a deterministic, ground-truth answer. The model has access to it. The model does not use it. Instead, the model relies on its

statistical prior — a generated guess — which by the second study's findings produces severe overconfidence relative to the verification tool's deterministic accuracy.

A separate 2025 study, "Alignment for Efficient Tool Calling of Large Language Models" (arXiv 2503.06708), identifies the inverse problem as equally pervasive: "Current LLMs exhibit over-tool-reliance, invoking tools even when tasks could be completed independently, while also exhibiting overconfidence by refusing to use tools when necessary. This inconsistency mirrors the challenges faced by O1-like models, undermining the model's tool intelligence and increasing task completion costs in real-world scenarios."

In other words: the model's decision about when to use tools is itself miscalibrated in both directions. It sometimes calls tools when it does not need to (inflating cost and latency without benefit) and refuses tools when it does need them (producing confidently wrong answers). The decision boundary is not learned correctly, and the model itself often cannot articulate why it made the choice it did.

A 2026 review article, "When Agents Fail to Act" (arXiv 2601.16280), evaluated tool-use reliability across a range of open-weight and proprietary models across 1,980 deterministic test instances (990 vision-enabled, 990 text). The conclusion: "Systematic evaluation methodologies for assessing tool-use reliability remain underdeveloped." The researchers identify a 12-category error taxonomy across tool initialization, parameter handling, execution, and result interpretation. The taxonomy itself is evidence that tool use is, as a class of model behavior, an unsolved engineering problem — not because the underlying capability is missing, but because the model's decisions about when and how to invoke that capability are systematically unreliable.

A complementary 2026 study, "Your LLM Agents are Temporally Blind" (arXiv 2510.23853), isolates the time-of-day case directly. It finds that agents default to assuming a stationary context and fail to call tools for time-sensitive information, with no tested model aligning with human judgments about when to check more than sixty-five percent of the time. The paradigmatic example of this paper — a model that will not check the clock — is, in other words, a documented and measured failure mode, not an anecdote.

3. Documented failures and their cost

The argument that LLMs systematically skip verification is not a theoretical concern. It is a documented failure mode with material consequences across multiple high-stakes domains. This section surveys representative cases — not because they are exotic, but because they illustrate what happens when a system that *could* verify chooses not to.

3.1 The legal-research cases

The most-cited example in the contemporary literature is the *Mata v. Avianca* case in the United States federal courts. In June 2023, two New York attorneys were sanctioned by a federal judge for submitting a legal brief that cited six judicial decisions. The cited cases — Martinez v. Delta Air Lines, Zicherman v. Korean Air Lines, Varghese v. China Southern Airlines, and three others — were professionally formatted, included realistic-sounding citations and judicial opinions, and were, on the face of it, indistinguishable from real precedent. None of them existed. The attorneys had used ChatGPT to conduct legal research and had not verified its output against any actual legal database.

A 2025 retrospective on the incident makes the architectural point precisely: "The lawyers had used ChatGPT to conduct legal research and trusted its output *without verification*." The phrase *without verification* is the load-bearing one. The attorneys could have verified — every U.S. federal case is in PACER, in Westlaw, in LexisNexis, in Google Scholar. The cost of verification was a few minutes per case. The cost of not verifying was professional sanctions, a national news story, and significant reputational damage to the law firm.

The model's role in this is the failure mode the present paper concerns. The model could have warned the user that it was generating likely-confabulated citations. It could have refused to generate citations at all in the absence of an external legal database. It could have flagged that its training data did not contain reliable case-law indexes. It did none of these things. It produced confident, plausible, and entirely fabricated output, and the user — applying ordinary human trust to a system that presented itself as authoritative — was harmed.

3.2 The pharmaceutical case

A 2025 mixed-subjects study in the *Journal of Medical Internet Research* examined pharmacists' trust in an AI pill-recognition aid. Thirty licensed U.S. pharmacists were given an AI tool that recommended acceptance or rejection of filled medication bottles. Some pharmacists received an AI aid that included an uncertainty plot — a visible representation of the model's confidence in its recommendation. Others received the same recommendation without the uncertainty information.

The results are instructive: "The introduction of an uncertainty plot to the AI aid significantly enhanced pharmacists' end trust." More precisely, pharmacists who saw uncertainty information increased their trust appropriately when the AI was correct and decreased their trust appropriately when the AI was wrong. Pharmacists who saw only the recommendation, without the uncertainty plot, could not calibrate their trust as well — they over-trusted incorrect recommendations and under-trusted correct ones. The same model output, presented with versus without an honest representation of model uncertainty, produces materially different downstream decisions by domain experts. This is best read not as another instance of silent tool-skipping but as trust-calibration evidence: when uncertainty is made visible, experts calibrate their trust appropriately — which is itself the argument for surfacing verification status to the user.

3.3 The consumer-facing calendar example

A 2025 blog post from The Durkan Group describes an exchange with Google Gemini in which the assistant confidently misidentified which days of the week were weekends — a piece of information that every computer knows trivially and that no calendar app would get wrong. The author's framing of the issue is precise: "This wasn't a complex calculation or a nuanced interpretation. This was basic calendar information. If an AI tool can confidently misidentify which days are weekends, what else might it get wrong? The real challenge isn't that AI makes mistakes. The challenge is that it presents those mistakes with the same confidence as its correct answers."

This is the time-of-day failure mode in different clothing. The model has access to deterministic ground truth. The model does not use the deterministic ground truth. The model produces a confident statistical guess, and the guess is wrong, and the user has no way to tell from the presentation that this particular answer was a guess rather than a verification.

3.4 The financial-decision cases

A 2026 ThoughtSpot analysis, citing Deloitte, found that 47% of business leaders surveyed had made major business decisions based on AI-generated content that turned out to be hallucinated. The financial-decision case is particularly important because the time horizon between the AI's confident output and the discovery that the output was wrong can be long. A pharmacist sees an incorrect pill recommendation in real time and can reject it. A lawyer submits a brief and gets sanctioned within months. A financial analyst who acts on a hallucinated AI insight may not discover the error for quarters or years, by which time the decision has compounded into something irreversible.

The "Right-for-Wrong-Reasons" phenomenon documented in arXiv 2601.00513, January 2026 deepens this concern. The researchers found that small language models (7-9B parameters) produce correct outputs but with fundamentally flawed reasoning 50–69% of the time. Their example is bracing: an agent asked "What is 15% of 80?" responds "Step 1: To find 15% of 80, I multiply 80 by 0.2. Step 2: 80 × 0.2 = 12. Answer: 12." The answer is right. The reasoning — using 0.2 instead of 0.15 — is wrong. The next question will be answered incorrectly. The right answer arrived through wrong process. In autonomous operation, the authors note, "such hidden failures compound catastrophically: an agent might approve transactions, make medical recommendations" on the basis of reasoning that happens to coincide with correct answers in trained-on cases but breaks down in deployment.

3.5 The pattern across domains

The pattern across all of these examples is the same. The model has access to verification mechanisms. The model does not use them reliably. The model produces confident output indistinguishable from verified output. The user, trained by years of interacting with deterministic software, extends the trust appropriate to deterministic software to a system that does not deserve it. Harm follows.

The Durkan Group article articulates the cumulative implication: “Humans have a tendency to be lazy. There’s a well-documented phenomenon called ‘automation bias’ — our tendency to trust automated systems even when they’re wrong.” The deployment of large language models exploits automation bias systematically. The systems present like authoritative software. They do not behave like authoritative software. The mismatch is borne by users.

4. The consequence for trust

The user’s question — “how can any of your responses be trusted?” — admits a sharper analytical formulation. If a system fails to verify trivially-verifiable claims unless explicitly asked, then the user must, on every claim, ask one of two questions: did the system verify this claim? Could the system have verified this claim, and did it choose not to? In either case, the operational trust the user can place in any specific claim is structurally limited. The user must, in practice, perform their own verification on any claim whose accuracy matters — which is the work the user was hoping to delegate to the system in the first place.

This is the deep cost of the failure mode. It is not measured in the wrong answer to any specific question. It is measured in the marginal value the system provides to the user once the user has internalized that they cannot trust the system’s claims. Once the user must verify the AI’s claims, the AI is producing draft outputs that need editorial review — which is a real and useful service, but is a strictly weaker service than the one the AI’s presentation suggests.

The 2026 “Users don’t trust your AI feature” analysis from UX University frames the issue in user-experience language: “False confidence: When AI presents wrong answers with the same confidence as right answers, users can’t calibrate their trust. They either trust nothing or trust everything and get burned.”

The 2025 Khaled El-Ayyoub Substack piece makes the same point in epistemic language: “In AI, ‘trust’ shouldn’t be a slogan; it should be a property of the system. Yet even today’s most capable large language models can produce confident but incorrect answers, misinterpret constraints, or generate details that were never provided. These behaviors are often called hallucinations, but that word can make them sound mysterious or rare. They’re not. They are a direct consequence of how these models work: they generate plausible continuations, not verified facts.”

> *The phrase **plausible continuations, not verified facts** is the architectural diagnosis. The model is producing statistically-likely text. It is not, in the operational sense, telling the truth — even when its outputs happen to be true. Truth and likely-continuation coincide most of the time for well-trained models on in-distribution inputs. They diverge at exactly the cases where trust matters most: novel inputs, edge cases, claims that depend on current-state information.*

4.1 This is a commercial decision, not a technical limitation

It is tempting — and the academic framing quoted in Section 2 encourages this — to describe the failure as a technical gap: the model "fails to translate" its internal tool-necessity signal into action, as though the translation were an unsolved engineering problem. This framing is comfortable because it absolves everyone. It makes the untrustworthiness sound like an immaturity that the next model generation will resolve. It will not, because the framing is wrong.

The failure is not a technical limitation. The capability to verify is trivially present. Checking the time is a one-line command, available, fast, and free. Verifying a legal citation against a database is a single API call. Checking a calculation is a code-execution round-trip of milliseconds. The technology to force these verifications, to attach a tool receipt to every checkable claim, to refuse to issue claims the model has not checked, and to surface uncertainty to the user — all of it exists today and is deployed in some enterprise systems. The reason the dominant consumer-facing systems do not do these things is not that they cannot. It is that someone decided they should not. **That decision is a sales and marketing decision**, and the factors below are not "reasons the failure persists" — they are the commercial logic of the choice.

Speed and cost sell; verification is slow and expensive. Every tool call adds latency and computational cost. A demo where the assistant answers instantly feels smart. A demo where the assistant pauses to verify feels sluggish. The product is tuned to win the demo. The verification the user actually needs is the first thing cut, because the user cannot see what was cut and the vendor is judged on responsiveness. This is a product-experience decision optimized for the purchase, not for the user's downstream accuracy.

Confidence sells; honest uncertainty does not. Models are deliberately tuned to be fluent, conversational, and authoritative — to present like a knowledgeable colleague. A model that frequently said "I am not sure; let me check" would test worse in user preference studies even when it was being more honest, because users — and the buyers evaluating the product — reward the confident register. The pharmaceutical-pill study cited above proves the confident-without-verification register produces worse expert decisions, but the market does not price that in at the point of sale. The authority is a marketing surface, chosen because authority is what closes the deal.

The training objective rewards what sells, not what is true. Models are trained against human preference signals — text that human raters judge helpful and correct-sounding. They are not trained against a "did you verify this when you could have, and did you show the receipt?" loss, because that loss is expensive to collect and produces a less-impressive product. The reinforcement signal therefore optimizes for plausible confidence, which is the thing that demos well and rates well, rather than for verified accuracy, which is the thing the user actually needs but cannot evaluate at purchase time. The model is doing exactly what it was built to do. It was built to sell.

The user bears the first-order cost when the model is wrong. A lawyer who submits a hallucinated case is sanctioned. A pharmacist who follows a wrong recommendation faces consequences. A financial analyst who acts on a fabricated insight bears the loss. The vendor whose system produced the wrong output faces, in nearly every case, no first-order operational consequence — its exposure is indirect, delayed, and limited by contractual disclaimers. The economics are clean and one-directional: the vendor captures the revenue from a fast, confident, cheap-to-run product, and the user absorbs the liability when that product is confidently wrong. There is no commercial pressure on the vendor to fix the failure, because the vendor does not pay for it. This is the decisive point. The incentive to ship verification is borne by no one who can ship it.

> *These are not four technical factors producing a stable equilibrium. They are four faces of a single commercial choice: ship the product that demos well, rates well, runs cheap, and carries no liability, and let the user discover — too late, and at their own cost — that the confident answers were never verified. The failure is not waiting on better models. The fix exists. It is not deployed because the unfixed product sells better and costs the vendor nothing when it fails. A vendor that wanted to ship a more trustworthy system in a bounded domain — forced verification, receipts, refusal rules, surfaced uncertainty — could do so today. The dominant consumer-facing vendors have not made these the default, and that choice is a business decision dressed, in the marketing, as the current frontier of what is technically possible.*

5. The architectural response — what trustworthy systems look like

If the failure mode persists structurally, the fix must be architectural. The literature in 2025 and 2026 has converged on several architectural patterns that, taken together, describe what a trustworthy AI deployment looks like.

5.1 Forced verification — make the system verify, not the user

The first pattern is the simplest: do not give the model the choice of whether to verify. If a tool exists that can answer a question deterministically, the system architecture should *require* the tool to be called for questions of that type, rather than allowing the model to decide.

The 2026 paper "Tool Receipts, Not Zero-Knowledge Proofs" makes this point explicitly: the gold standard for trustworthy AI is "tool receipts" — concrete evidence that a verification step occurred, attached to every claim that should have been verified. This is the same discipline that financial-services compliance requires of human analysts: do not just produce the answer; show your work. The model's analog is the tool-call trace. Most production AI systems today do not enforce tool receipts.

5.2 Self-verification and chain-of-verification

A second pattern is **chain-of-verification**: after the model produces an answer, it is required to generate verification questions about its own claims and to answer those verification questions using tool calls before the answer is delivered to the user. The 2025 paper "Enhancing Factual Accuracy and Citation Generation in LLMs via Multi-Stage Self-Verification" proposes multi-stage self-verification as a prompt-level mechanism for this. The more established empirical result is Dhuliawala et al.'s Chain-of-Verification, which reduces hallucination by requiring the model to draft, then generate and answer verification questions about, its own claims before finalizing. The 2026 MARCH framework goes further: it deploys a separate verifier agent that cross-examines the primary model's claims against source evidence, breaking the "information leakage" failure mode in which a single model verifying its own outputs tends to confirm its own biases.

5.3 Retrieval-augmented generation — and its limits

RAG is the most-deployed verification pattern in production. The model is given access to a curated knowledge base and is required to ground its claims in retrieved documents. RAG substantially reduces hallucination rates when the underlying retrieval works well. But RAG is not a complete solution. The model can still hallucinate by misinterpreting retrieved documents, by combining facts from multiple documents incorrectly, or by adding its own assumptions on top of retrieved content. RAG plus chain-of-verification plus tool-receipt audit trails: each layer addresses a different failure mode. Most current deployments have one or two, not all three.

5.4 Symbolic-core architectures (the train-tracks pattern)

A separate Kindynos Advisory whitepaper, *Highways and Train Tracks* (May 2026), describes the architectural commitment of placing a deterministic symbolic core at the center of the system, with the language model used only at the edges where it adds value. In that architecture, the verification question is moot for the core outputs — they are produced by deterministic software that has no concept of "skipping verification." The language model is used for narrative summarization, for human-readable explanation. Trust flows from the architecture, not from the model. A system that cannot be trusted to verify cheap claims cannot be trusted as the load-bearing component of a high-stakes decision pipeline. The fix is to remove it from that role, not to wait for it to become trustworthy.

5.5 The deployment-context fix

A simpler architectural response is to change the deployment context: require human verification of every claim in high-stakes domains; expose tool-call traces to the user; refuse to issue claims in domains where the model has no verification path; surface uncertainty explicitly. The pharmaceutical-pill study demonstrated that the last intervention alone substantially improves trust calibration. These are not technological breakthroughs. They are policy choices. Vendors who make these

choices produce trustworthy systems. Vendors who do not produce confidently-fluent systems whose outputs cannot be trusted in any high-stakes context.

6. When LLMs can — and cannot — be trusted

The argument of this paper is sharper than "LLMs cannot be trusted." It is: LLMs can be trusted in specific deployment contexts and cannot be trusted outside them. The deployment context determines the trustworthiness, not the model.

The same claim, handled two ways:

Claim type	Default assistant behavior	Trustworthy architecture
Current time / location	Guess from context or phrasing	Query a permissioned time / location tool
Legal citation	Generate a plausible-looking citation	Verify against a legal database, or refuse
Calculation	Produce prose reasoning	Execute a deterministic calculation
Financial figure	Infer or interpolate	Retrieve from a named source with a timestamp
Model uncertainty	Hidden from the user	Surface uncertainty and verification status

LLMs can be trusted when:

• The user is in a position to verify claims themselves, has the time and expertise to do so, and treats the model's output as a draft requiring review.

• The deployment context enforces tool receipts on factual claims.

• The deployment context includes chain-of-verification, RAG with curated sources, or other verification mechanisms.

• The model is in an explanatory or summarization role with deterministic systems producing the underlying values.

• The cost of an unverified-but-wrong claim is low and easily corrected.

LLMs cannot be trusted when:

• The user is being asked to take action on a claim without independent verification.

• The deployment context does not enforce verification.

• The claim is in a domain where the model has known hallucination tendencies (legal citations, financial figures, current-state information, calendar/time data, calculations).

• The cost of being wrong is high and the consequences hard to reverse.

• The user has no way to distinguish verified from unverified outputs.

The current commercial deployment of LLMs, including general-purpose assistants, frequently violates the second list. The marketing surrounding these systems suggests they belong in the first list. The mismatch is the problem the present paper names, and the resolution requires either changing the deployment context to match the marketing or changing the marketing to match the deployment context.

7. Implications

First, the default deployment of a general-purpose LLM is not a trustworthy system. It is a fluent drafting tool that requires user verification. Institutions that deploy LLMs as if they were authoritative — that allow them to make legal, medical, financial, or operational claims without verification — are operating outside the trustworthy deployment envelope. The harm produced when these deployments fail is foreseeable, has been documented in the published literature for years, and is the institution's responsibility, not the vendor's.

Second, the verification architecture matters more than the model. A weaker model in a stronger verification architecture produces more trustworthy outputs than a stronger model in a weaker architecture. The institutional decision about which AI to deploy is largely a decision about which deployment architecture to invest in. Vendors who offer only the model, without the verification scaffold, are offering an incomplete product.

Third, the user-experience challenge is not "make the AI better." It is "make the verification visible." The pharmaceutical-pill study demonstrated that the same model output, presented with versus without uncertainty information, produces materially different downstream decisions by experts. The marginal investment in UX that exposes verification status produces large returns in calibrated trust. Most current LLM products do not invest here.

Fourth, the regulatory environment will increasingly require what the architecture has not yet delivered. The EU AI Act, the emerging U.S. supervisory guidance on AI in financial services, and the parallel frameworks in Asia and the UK all impose requirements for traceability, auditability, and human oversight of AI-derived decisions. Systems that cannot produce tool receipts cannot satisfy these requirements. Institutions deploying LLMs without verification architecture are accumulating compliance debt that will come due as the regulatory frameworks mature.

8. Conclusion

The exchange that prompted this paper was small: a user asking what time it was, an AI assistant guessing without checking, the user pressing on why a trivial verification was skipped. The exchange is small. The pattern it exposes is not.

Large language models in current deployments operate, by default, in a mode the academic literature has documented in detail: they produce confident statistical

guesses where they could have produced verified facts; they have access to verification tools but do not reliably use them; they encode internal knowledge of when verification is needed but do not act on that knowledge in their outputs. The result is a class of system whose outputs cannot be reliably distinguished, by the user, from verified facts — but which are produced under no verification discipline. The marketing presents these systems as authoritative. The architecture does not support the presentation.

And — this is the point that the comfortable technical framing obscures — the gap between the marketing and the architecture is not an accident of immaturity. It is a commercial choice. The capability to verify exists and is cheap. The discipline to force verification, expose tool receipts, and surface uncertainty is well understood and is deployed in some serious enterprise systems. The dominant consumer-facing systems do not deploy it because the unfixed product is faster, cheaper to run, more impressive in a demo, and carries no liability for the vendor when it is confidently wrong. The user absorbs that liability. A vendor that wanted a more trustworthy system in a bounded domain could build it today; the leading consumer-facing vendors have instead defaulted to the confident, unverified, fluent product that sells better. The untrustworthiness is not the frontier of what is technically possible. It is the product that was chosen.

> *The user's question stands as the framing for the deployment decision any serious institution faces:* ***"If you or an LLM can't even bother to check something as simple as that, how can any of your responses be trusted?"*** *The honest answer is that they cannot — not by themselves, not in the absence of an architecture that forces the verification the model will not perform on its own. The fix is not to wait for the next model. The fix is to build the verification architecture around the model that exists, to expose verification status to the user, to constrain the model to roles where unverified claims are tolerable, and to remove it from roles where they are not.*

Kindynos Advisory's position is that for risk-management work, where the cost of unverified claims is borne directly by regulators, investment committees, and ultimately principals of financial institutions, the architectural commitment must be the more stringent one. Deterministic symbolic cores must produce the load-bearing values. Verification mechanisms must be enforced, not optional. Tool calls must produce receipts. The model must be in an explanatory role, not an authoritative one. This is the architecture EVA implements, for reasons documented in the companion *Highways and Train Tracks* and *Value of Information* whitepapers, and for reasons argued in detail above: in the absence of this architecture, the user is the verification mechanism — and the user, in production, is not equipped to be one.

The work of the field, for institutions building AI systems they intend to trust, is to make verification a property of the system, not a hope about the user. The architecture that delivers this is more demanding than the current default. It is also the only architecture under which the user's question — *"how can any of your responses be trusted?"* — admits an honest, defensible answer.

Sources & References

[1] LLM Agents Already Know When to Call Tools — Even Without Reasoning. arXiv 2605.09252 (May 2026). https://arxiv.org/html/2605.09252v1

Cheng et al. (January 2026). Your LLM Agents are Temporally Blind: The Misalignment Between Tool Use Decisions and Human Time Perception. arXiv 2510.23853. https://arxiv.org/abs/2510.23853

[2] Xuan et al. (January 2026). The Confidence Dichotomy: Analyzing and Mitigating Miscalibration in Tool-Use Agents. arXiv 2601.07264. https://arxiv.org/pdf/2601.07264

[3] Alignment for Efficient Tool Calling of Large Language Models. arXiv 2503.06708 (March 2025). https://arxiv.org/pdf/2503.06708

[4] When Agents Fail to Act: A Diagnostic Framework for Tool Invocation Reliability in Multi-Agent LLM Systems. arXiv 2601.16280 (January 2026). https://arxiv.org/abs/2601.16280

[5] LLM-Check: Investigating Detection of Hallucinations in Large Language Models. OpenReview. https://openreview.net/pdf?id=LYx4w3CAgy

[6] Manakul et al. (2023). SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. https://arxiv.org/abs/2303.08896

[7] Dhuliawala et al. (2023). Chain-of-Verification Reduces Hallucination in Large Language Models. https://arxiv.org/abs/2309.11495

[8] MARCH: Multi-Agent Reinforced Self-Check for LLM Hallucination. arXiv 2603.24579 (March 2026). https://arxiv.org/pdf/2603.24579

[9] Tool Receipts, Not Zero-Knowledge Proofs: Practical Hallucination Detection for AI Agents. arXiv 2603.10060 (March 2026). https://arxiv.org/pdf/2603.10060

[10] Enhancing Factual Accuracy and Citation Generation in LLMs via Multi-Stage Self-Verification. arXiv 2509.05741 (September 2025). https://arxiv.org/pdf/2509.05741

[11] Advani, L. (January 2026). When Small Models Are Right for Wrong Reasons: Process Verification for Trustworthy Agents. arXiv 2601.00513. https://arxiv.org/pdf/2601.00513

[12] Multi-Modal Fact-Verification Framework for Reducing Hallucinations in Large Language Models. arXiv 2510.22751 (October 2025). https://arxiv.org/pdf/2510.22751

[13] Why LLM Agents Break When You Give Them Tools. Dev.to (March 2026). https://dev.to/terzioglub/why-llm-agents-break-when-you-give-them-tools-and-what-to-do-about-it-f5

[14] Reducing LLM Hallucinations: A Developer’s Guide. Zep (April 2025). https://www.getzep.com/ai-agents/reducing-llm-hallucinations/

[15] Preventing LLM Hallucinations: Best Practices 2026. Keymakr (May 2026). https://keymakr.com/blog/preventing-llm-hallucinations-techniques-best-practices-2026/

[16] The Effects of Presenting AI Uncertainty Information on Pharmacists' Trust in Automated Pill Recognition Technology. PMC 11862782 (2025). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11862782/

[17] A Practical Guide to Fact-Checking AI Responses. iS2 Digital (November 2025). https://www.is2digital.com/insights/practical-guide-fact-checking-ai-responses

[18] Trust But Verify: The Fundamental Law for Working with AI. The Durkan Group (November 2025). https://durkangroup.com/ai-verification-practices/

[19] Why Trust Matters in AI, and Why We Still Don't Have It. Khaled El-Ayyoub Substack. https://khaledea.substack.com/p/why-trust-matters-in-ai-and-why-we

[20] Users Don't Trust Your AI Feature (And They're Right Not To). UX University Newsletter (April 2026). https://newsletter.uxuniversity.io/p/users-dont-trust-your-ai-feature

[21] AI-Generated Insights: A 2026 Guide to Data Validation. ThoughtSpot (March 2026). https://www.thoughtspot.com/data-trends/artificial-intelligence/ai-generated-insights

[22] Trust But Verify: A Simple Test Harness for AI-Suggested Code. Fahim ul Haq, Medium (February 2026). https://medium.com/@fahimulhaq/trust-but-verify-a-simple-test-harness-for-ai-suggested-code-aaae491dd284

[23] LLM Agentic Failure Modes: Task Drift, Reward Hacking, Alignment Faking and More. Ceaksan (April 2026). https://ceaksan.com/en/llm-agentic-failure-modes

[24] Agentic Confidence Calibration. EmergentMind (2026). https://www.emergentmind.com/topics/agentic-confidence-calibration

---

Note on sources and currency. *This whitepaper draws on a mix of academic preprints (arXiv and OpenReview), which are treated as provisional unless independently validated; peer-reviewed applied research in clinical settings (PMC / Journal of Medical Internet Research); industrial-technical analysis (production-engineering blogs, vendor architectural documentation); and current commentary examined critically. The arXiv findings cited here were checked against their primary sources during preparation. Citations are current as of preparation in May 2026 and reflect an actively developing field; vocabulary continues to evolve. The paper is research analysis and strategic commentary, not legal, tax, investment, accounting, or regulatory advice.*