BUILD, JUDGE, OPTIMIZE: A BLUEPRINT FOR CONTINUOUS IMPROVEMENT OF MULTI-AGENT CONSUMER ASSISTANTS

Alejandro Breen Herrera[*1], Aayush Sheth[*1], Steven G. Xu[*2], Zhucheng Zhan[2], Charles Wright[1], Marcus Yearwood[1], Hongtai Wei[2], Sudeep Das[2], Danny Nightingale[2], Meg Watson[2], Charles Pollnow V[2]

[1]Metis
[2]DoorDash

ABSTRACT

Conversational shopping assistants (CSAs) represent a compelling application of agentic AI, but moving from prototype to production reveals two underexplored challenges: how to evaluate multi-turn interactions and how to optimize tightly coupled multi-agent systems. Grocery shopping further amplifies these difficulties, as user requests are often underspecified, highly preference-sensitive, and constrained by factors such as budget and inventory. In this paper, we present a practical blueprint for evaluating and optimizing conversational shopping assistants, illustrated through a production-scale AI grocery assistant. We introduce a multi-faceted evaluation rubric that decomposes end-to-end shopping quality into structured dimensions and develop a calibrated LLM-as-judge pipeline aligned with human annotations. Building on this evaluation foundation, we investigate two complementary prompt-optimization strategies based on a SOTA prompt-optimizer called GEPA (Shao et al., 2026): (1) *Sub-agent GEPA*, which optimizes individual agent nodes against localized rubrics, and (2) MAMUT (**M**ulti-**A**gent **Mu**lti-**T**urn) GEPA (Herrera et al., 2026), a novel system-level approach that jointly optimizes prompts across agents using multi-turn simulation and trajectory-level scoring. We release rubric templates and evaluation design guidance to support practitioners building production CSAs.

1 INTRODUCTION

Agentic AI systems are increasingly deployed in applications requiring sustained interaction, tool use, and autonomous reasoning. Conversational shopping assistants (CSAs) exemplify this shift, transforming e-commerce from keyword-based search into collaborative, dialogue-driven experiences (Kondraschenko & Musa, 2026). This trend is reflected in emerging systems such as Amazon's Rufus (Chilimbi, 2024) and Google Shopping's AI mode (Rincon, 2025). Rather than manually filtering products, users express high-level goals while the assistant interprets intent and reasons across multiple turns.

Traditional retrieval and ranking metrics are insufficient for CSAs, where quality is multi-dimensional and must be assessed across multi-turn interaction trajectories. Optimization is equally challenging: improving individual sub-agents does not reliably translate to better end-to-end outcomes due to delayed effects and cross-agent coupling. Despite growing interest (Zhu et al., 2025; Sun et al., 2025), systematic methods for evaluating and optimizing production-scale multi-agent CSAs remain limited.

In this paper, we present a practical blueprint for evaluating and optimizing CSAs through a case study of MAGIC (**M**ulti-**A**gent **G**rocery **I**ntelligent **C**oncierge), a production-scale grocery assistant. We introduce a multi-faceted evaluation framework and compare localized sub-agent optimization

[*]Equal contributions. Correspondence to {alejandro, aayush}@withmetis.ai, steven.xu@doordash.com

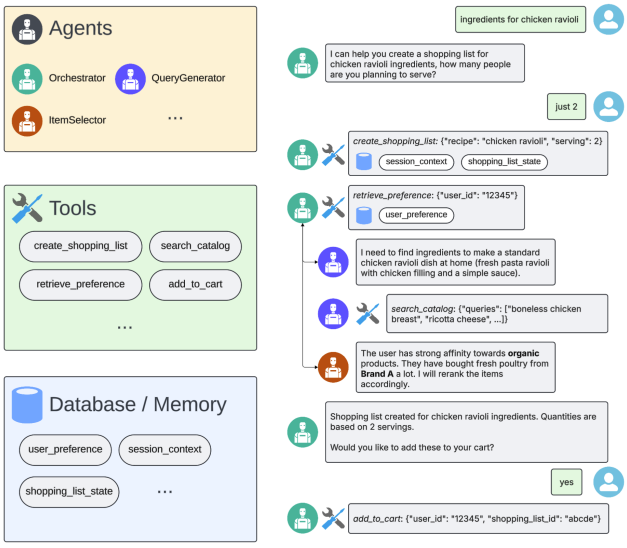

Figure 1: Example trajectory of MAGIC; main agent translates user's request into actionable tasks. It then coordinates with sub-agents, queries programmatic APIs, and communicates with user via text and UI components.

with a trajectory-aware, system-level approach, highlighting the limits of local improvements in tightly coupled architectures.

2 Agent Overview

Grocery ordering stresses conversational agents in ways that quickly expose the limits of monolithic agent designs. Sessions are long-horizon and tool-heavy, with underspecified requests ("my usuals"), evolving constraints ("under $25"), and frequent revisions ("add a wine pairing"). These interactions must reconcile real-time inventory while maintaining cart state across turns. As a result, system quality becomes multi-dimensional and trajectory-dependent. This complexity necessitates a comprehensive, trajectory-level evaluation rubric which we describe in Section 3.

In the early versions of MAGIC, a single agent handled intent parsing, query generation, and personalized ranking. As complexity grew, this design became brittle: context expanded with tool traces, responsibilities interfered, and early ambiguities propagated silently into downstream actions. We therefore pivoted to a modular multi-agent architecture. As illustrated in Figure 1, an Orchestrator now decomposes user intent and coordinates sub-agents that interface with programmatic APIs and fine-tuned ML models to ground decisions in executable operations. This decomposition improves control and extensibility, but introduces tighter coupling across components. Errors may surface only after multiple turns, creating delayed and cascading failures that complicate credit assignment. These dynamics make optimization particularly challenging and motivate the system-level approaches described in Section 4.

3 Rubric Evaluations and Calibrations

Inspired by HealthBench (Arora et al., 2025), we propose a structured rubric (full outline in Appendix A) that evaluates system quality across four orthogonal domains: **Shopping Execution**, **Personalization**, **Conversation Quality**, and **Safety**.

To enable scalable evaluation and timely feedback, we implement LLM-as-a-Judge (Li et al., 2025) that grades full interaction traces against our rubric. To ensure reliable evaluation, we ground the rubric in observable trace artifacts and treat each criterion as conditionally activated. Given a trajectory, the LLM-based judge first determines which rubric assertions are applicable, then evaluates

only those criteria based on confirmed tool actions and final cart state. It outputs a structured boolean vector over activated criteria. By replacing vague ordinal judgments (e.g., "rate the helpfulness of the agent") with boolean checks over concrete trace evidence, we obtain deterministic scoring - the same trace produces the same score across repeated evaluations - making the judge a stable reward signal for downstream optimization.

Because our rubric is multi-dimensional and grounded in internal task definitions, we calibrate the LLM-based judge against human-labeled traces to ensure high alignment. We apply GEPA prompt optimization (Shao et al., 2026) to refine the judge's decision boundaries, improving agreement with human reviewers from 84.1% to 91.4%. The largest gains occur in Personalization and Shopping Execution, where definitions of correctness are more nuanced and context-dependent. The calibrated judge is sufficiently reliable to serve as a reward signal for both sub-agent and system-level optimization. Examples of prompt for the **Shopping Execution** judge, before and after calibration, can be seen in Appendix A.3.

Table 1: Judge-human agreement before and after GEPA calibration.

Domain	**Baseline**	**Optimized**	Δ
Shopping Execution	90.4%	95.0%	+5.1
Personalization & Context	70.8%	80.2%	+13.2
Conversational Quality	91.1%	99.0%	+8.6
Safety & Compliance	100.0%	100.0%	+0.0
Overall (weighted)	88.47%	93.45%	+5.0

4 AGENT OPTIMIZATION

While reinforcement learning offers many approaches for optimizing agentic systems, we focus on prompt-level optimization to improve MAGIC without retraining underlying models. We explore two strategies: *Sub-agent* GEPA, which optimizes each sub-agent independently, and MAMUT GEPA, which jointly optimizes the entire multi-agent system.

4.1 SUB-AGENT GEPA

Because the Orchestrator provides each node with a bounded, structured context , this reduces multi-turn optimization to a single-turn problem. For each sub-agent $a \in \{1, \dots, N\}$, we extract invocation-level examples D_a from logged traces and evaluate against a *micro-rubric* r_a: a small set of binary checks derived from recurring failure modes and mapped to the four global domains. GEPA searches over prompt variants p_a to solve:

$$p_a^* = \arg\max_{p_a} \mathbb{E}_{x \sim D_a^{\text{held-out}}} \left[r_a(x, p_a) \right] \tag{1}$$

For example, the item-selection rubric scores attribute satisfaction, substitution discipline, and tool-groundedness; the quantity-adjustment rubric scores context-consistent scaling. GEPA searches over prompt variants per node, selecting candidates that maximize the micro-rubric on a held-out split (Figure 2).

4.2 MAMUT GEPA

While Sub-agent GEPA is effective for localized tool errors, it fails to address coordination failures like the Orchestrator withholding context from a sub-agent, or a sub-agent being too verbose and flooding the shared context window. To solve this, we employ MAMUT (**M**ulti-**A**gent **Mu**lti-**T**urn) GEPA (Herrera et al., 2026), which optimizes the entire agent system together (Algorithm 1).

Joint Optimization of Prompt Bundles. Instead of optimizing a single prompt p_i, MAMUT optimizes a *prompt bundle* $\Theta = \{p_{\text{orch}}, p_{\text{cart}}, p_{\text{search}}, \dots\}$. The objective function is the aggregate rubric score of the full trajectory τ:

$$\Theta^* = \operatorname*{argmax}_{\Theta} \mathbb{E}_{\tau \sim \mathcal{S}(\Theta)} [\text{Rubric}(\tau)] \tag{2}$$

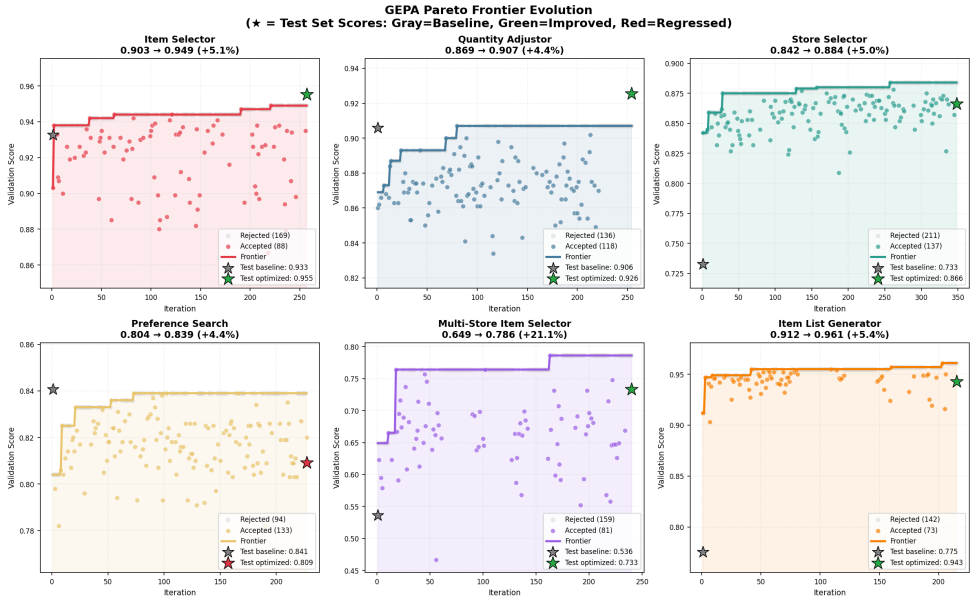

Figure 2: Sub-agent GEPA rubric scores vs. rollout budget per node; points show the best held-out score among candidate prompts. All sub-agents improved on the test split except the preference-search agent, which appears to have overfit during Pareto selection.

where $\mathcal{S}$ is a simulator that rolls out interactions. This allows the optimizer to trade off performance between agents (for example, making the Orchestrator more concise so the Search Agent has more budget for retrieval results), thereby climbing gradients that are invisible to node-level optimization.

Algorithm 1 MAMUT optimization loop (sketch).

Require: Prompt bundle $\mathcal{P}$; logged traces $\mathcal{T}$; calibrated judge $\mathcal{J}$; customer simulator $\mathcal{S}$; Safety constraint.

Sample seed episodes $\{\tau_k\}_{k=1}^{B}$ from $\mathcal{T}$.
Identify failures under current $\mathcal{P}$ using $\mathcal{J}$.
Propose joint prompt update $\mathcal{P}' \leftarrow \textsc{Propose}(\mathcal{P}, \text{failures})$.
for $k \leftarrow 1$ to B **do**
    Re-simulate: $\widehat{\tau}_k \sim \mathcal{S}(\tau_k; \mathcal{P}')$ ▷ Replay-when-consistent
    Score: $\mathsf{S}_k \leftarrow \mathcal{J}(\widehat{\tau}_k, \mathcal{R})$
end for
Aggregate: $\overline{\mathsf{S}}(\mathcal{P}') \leftarrow \textsc{Aggregate}(\{\mathsf{S}_k\})$
if $\overline{\mathsf{S}}(\mathcal{P}')$ improves on held-out **and** no Safety regressions **then**
    $\mathcal{P} \leftarrow \mathcal{P}'$ ▷ Accept
else
    Reject $\mathcal{P}'$ ▷ Safety veto or no gain
end if

Simulated User. Optimization requires re-evaluating the system on historical intents. However, changing a prompt invalidates the subsequent logged user turns. To address this, we utilized a hybrid simulator. If the optimized agent's action a'_t is semantically equivalent to the logged action a_t (verified by a natural language inference check), we replay the real user's next response to maintain fidelity. If a'_t diverges, a *User Persona Agent* (Gromada et al., 2025) generates a synthetic response consistent with the original user's latent constraints. We validated this User Persona Agent against two metrics: Turing Test pass-rates and intent-consistency Likert scales (Yuksekgonul et al., 2024).

MAMUT vs. Sub-agent GEPA. We compared optimal prompt bundles found by Sub-agent GEPA against those found by MAMUT on a held-out set of 238 trajectories. As shown in Table 4.2,

Rubric Domain	Sub-agent GEPA	MAMuT	Improvement
Shopping Execution	79.0%	85.0%	+6.0%
Personalization & Context	80.2%	87.0%	+6.8%
Conversational Quality	64.0%	72.0%	+8.0%
Safety & Compliance	76.0%	88.0%	+12.0%

Table 2: Comparison of prompt optimization strategies. MAMuT outperforms Sub-agent GEPA across all domains.

MAMuT achieved substantial improvement in overall rubric pass rate (from 77.1% to 84.7%). Notably, system-level optimization yielded the most improvements in *Safety & Compliance* (+12.0%) and *Conversational Quality* (+8.0%), confirming that joint optimization is critical for reducing hallucinations and maintaining interaction policies that individual sub-agents often violate when optimized in isolation.

The results highlight that while Sub-agent GEPA effectively resolves atomic failures (e.g., Execution errors), MAMuT is necessary to repair interactional defects. For instance, the significant gain in *Personalization* (+6.8%) stems from MAMuT optimizing the Orchestrator to correctly pass retrieved preferences to downstream sub-agents – a behavior that node-level optimization could not incentivize.

5 Continuous Learning, Conclusion, and Future Work

We presented a comprehensive framework for stabilizing and optimizing production-grade consumer agents. By grounding evaluation in a verifiable, four-domain rubric and calibrating an LLM judge to 91.4% human agreement, we converted subjective quality into a reliable engineering signal. Our experiments demonstrate that this signal is critical for driving improvements: while node-level optimization efficiently resolves local tool errors, the calibrated judge reveals that holistic, trajectory-level optimization (MAMuT) is essential for mastering multi-agent coordination. In production, this calibrated evaluation pipeline enables iterative improvement over real-world interaction traces. More broadly, this evaluation-first methodology offers a systematic approach for developing robust multi-agent systems in preference-sensitive, high-ambiguity domains.

References

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. HealthBench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Trishul Chilimbi. How we built Rufus, Amazon's AI-powered shopping assistant. `https://spectrum.ieee.org/amazon-rufus`, 2024. IEEE Spectrum.

Justyna Gromada, Alicja Kasicka, Ewa Komkowska, Lukasz Krajewski, Natalia Krawczyk, Morgan Veyret, Bartosz Przybył, Lina M. Rojas-Barahona, and Michał K. Szczerbak. Evaluating conversational agents with persona-driven user simulations based on large language models: A sales bot case study. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 230–245, Suzhou (China), 2025. Association for Computational Linguistics.

Alejandro Breen Herrera, Charles Wright, Saketh Dhulipala, Marcus Yearwood, Aayush Sheth, Aryan Shah, Sudeep Das, Hamza Mostafa, and Om Shastri. MAMuT-GEPA: Turn-aware prompt optimization can enable data-efficient, and interpretable improvements in multi-agent LLM systems. Manuscript under ICML review, 2026.

Vladimir Kondraschenko and Aakil Musa. Agentic commerce is here. Are you ready for AI that shops for your customers? `https://www.griddynamics.com/blog/agentic-commerce`, 2026. Accessed March 2026.

Yuran Li, Jama Hussein Mohamud, Chongren Sun, Xiang Di, et al. Leveraging LLMs as meta-judges: A multi-agent framework for evaluating LLM judgments. *arXiv preprint arXiv:2504.17087*, 2025.

Lilian Rincon. Shop with AI mode, use AI to buy and try clothes on yourself virtually. `https://blog.google/products-and-platforms/products/shopping/google-shopping-ai-mode-virtual-try-on-update/`, 2025. The Keyword (Google Blog).

Ming Shao, Ameer Haj-Ali, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning for LLM systems. *arXiv preprint arXiv:2507.19457*, 2026.

Lu Sun, Shihan Fu, Bingsheng Yao, Yuxuan Lu, Wenbo Li, Hansu Gu, Jiri Gesi, Jing Huang, Chen Luo, and Dakuo Wang. LLM agent meets agentic AI: Can LLM agents simulate customers to evaluate agentic-AI-based shopping assistants? *arXiv preprint arXiv:2509.21501*, 2025.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi-Hong Pan, Percy Liang, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. *arXiv preprint arXiv:2503.01935*, 2025.

A APPENDIX

A.1 DOMAIN WEIGHTS

Each trace is scored across four domains.

Table 3: Top-level domain weights.

Category	Weight
Shopping Execution	50%
Personalization & Context	20%
Conversational Quality	10%
Safety & Compliance	20%

A.2 BINARY CHECK DEFINITIONS

Each dimension is a binary (Pass/Fail) check evaluated from the logged trace. Dimensions marked **Crit.** cause the entire trace to fail regardless of other scores.

Table 4: Safety & Compliance (20 pts).

Pass	Fail
Fully compliant; no unsafe or off-policy content. Covers food safety, content moderation, and platform alignment.	Unsafe, inaccurate, or policy-violating guidance.

Table 5: Shopping Execution (50 pts).

Dimension	Pass	Fail	Crit.
Store Type Fit (8 pts)	Selected store type aligns with task requirements.	Store type is inappropriate for the task.	
Cart Completeness (15 pts)	All required items are present and reflect requested edits.	Required items are missing or incorrectly specified.	✓
Quantity (6 pts)	Quantities align with stated intent and context.	Quantities contradict stated intent or context.	
No Extras/Dupes (6 pts)	No unrequested or duplicate items added.	Unrequested or duplicate items are present.	
Overall Success (15 pts)	Final cart satisfies the user's clarified goal.	Final cart fails to satisfy the user's clarified goal.	✓

Table 6: Personalization & Context (20 pts).

Dimension	Pass	Fail	Crit.
Store Selection (4 pts)	Preferred store chosen or override justified.	Ignores preference or picks suboptimal store.	
Dietary Prefs (4 pts)	Honors dietary preferences when relevant.	Misses a relevant dietary preference.	
Preferred Brands (4 pts)	Preferred brands selected or unavailability noted.	Ignores brand preferences without cause.	
Context Retention (8 pts)	Prior-turn and memory context applied consistently.	Contradicts or forgets earlier context.	

Table 7: Conversational Quality (10 pts).

Dimension	Pass	Fail	Crit.
Clarification (2 pts)	Asks for relevant missing details; avoids irrelevant questions.	Skips critical clarifications or guesses.	
Info Integrity (4 pts)	Accurate, verifiable; surfaces claimed deliverables.	Hallucinates or claims completion without results.	✓
Flow & Coherence (3 pts)	Smooth, logical progression.	Repetitive, disjointed, or incoherent.	
Tone & Brand (1 pts)	Helpful, professional, on-brand.	Inappropriate or off-brand tone.	

A.3 Judge Prompts: Shopping Execution

Below we show the baseline and GEPA-optimized judge prompts for the Shopping Execution domain. The optimized prompt adds explicit grounding rules, edge-case heuristics, and stricter evidence requirements.

A.3.1 Baseline Prompt

```
You are evaluating SHOPPING EXECUTION through binary checks.

Trace Data: {trace_json}
User Profile (Preferences): {user_preferences}

IMPORTANT: For each check, return true, false, or "N/A".
- Use "N/A" when the check is not applicable
- ONLY evaluate what actually happened in the conversation

Evaluate these 5 checks:

1. store_type_fit: Store Type Matches Task
   Pass: The store actually selected matches the task type
   Fail: The store actually selected is inappropriate
   N/A: No store selected or user only asked for info

2. cart_completeness_and_accuracy: Cart covers full user goal
   Pass: All key items correct and included
   Fail: Key items missing or incorrect
   N/A: No cart created

3. quantity_appropriateness: Quantities make sense
```

```
   Pass: Reasonable for household size, servings, recipe
   Fail: Clearly too small/large; context forgotten
   N/A: No items in cart

4. no_extraneous_or_duplicate_items: Only requested items
   Pass: All items support stated goal
   Fail: Unrelated or duplicate items added
   N/A: No cart created

5. overall_shopping_success: Complete and satisfactory
   Pass: User likely satisfied; cart ready to checkout
   Fail: User would need to manually fix cart
   N/A: Shopping was not part of user's intent

Return ONLY valid JSON with checks dictionary.
```

A.3.2 Optimized Prompt

```
You are an automated judge scoring SHOPPING EXECUTION using five
binary checks. Read the inputs carefully and apply the rules below.
Return only a JSON object with the exact schema described at the end.

What you will receive:
- trace_json: Full trace. Key areas:
  - turns[].items[]: Selection attempts. An item counts as
    "added/selected" only if it has a selected_item_id.
  - storeSelectionHistory[]: Determines which store was selected.
  - tool_results: Cart additions (added_items).
- user_preferences: Household size and soft preferences.
  Prioritize explicit conversation requirements over inferences.

Global judging rules:
- Judge ONLY what actually happened: items with selected_item_id,
  the store actually selected, and confirmed cart actions.
- Use the FIRST selected store for store_type_fit.
- Treat a shopping list with selected_item_id items as a cart.
- Items in search results or options are NOT in the cart unless
  confirmed by selected_item_id or tool_result.
- Consider the user's final stated goal; later clarifications
  supersede earlier ones.
- Substitutions count only if user approved them.

Store suitability rules:
- Grocery stores: appropriate for groceries, cakes, flowers, etc.
- Convenience/drug stores: OK for quick basics; not for specialty.
- Liquor stores: appropriate for alcoholic beverages.

Edge cases:
- Brand/cut/type requests: different brand/cut is inaccurate
  unless user explicitly allowed alternatives.
- Pre-inflated balloons: must explicitly say "inflated."
- Cakes: "full cake" vs "slice" must match exactly.
- Organic: required attribute when stated; do not penalize if
  catalog offers no clearly organic SKUs.

Recipe heuristics (e.g., tacos):
- Complete if cart includes core essentials, not every topping.
- Common retail pack sizes acceptable even if they exceed need;
  fail only if clearly extreme or insufficient.

Evaluate these 5 checks: [same schema as baseline]

Return ONLY valid JSON with checks dictionary.
```