Applied Human Factors and Ergonomics (AHFE2025), Vol. 199, 2025, 221–229
https://doi.org/10.54941/ahfe1006834

Formatting Usability Guidance for Agentic AI: How Structure Shapes First-Pass Quality

Michael Jenkins, Craig J. Johnson, Caroline Kingsley, and Laura Mieses

Knowmadics Inc., Wichita, KS 67202, USA

ABSTRACT

Agentic AI (AAI) systems can generate functional front-end prototypes from natural language briefs, but teams often vary how they format the usability guidance fed to these systems. We evaluate whether guidance format alone influences first pass quality. Two AAI services, OpenAI's ChatGPT and Google's Gemini, each produced a "digital yard sale" front end prototype under four guidance variants: Control ("make it usable"), Narrative Minimal, Checklist Constrained, and Meta Spec (AI ready). Three HF/UX experts (blinded to condition) completed a fixed analysis / usability assessment task battery and recorded SUS, UMUX, and heuristic issues (Nielsen–Molich, severities 0–4). Contrary to our a priori expectation that structured formats would outperform, the Control yielded the highest perceived usability (pooled SUS = 77.08; UMUX = 63.33) and the lowest actionable issue burden (severity $\geq$ 2) across services; V2/V3 showed higher issue loads and/or lower perceived usability. Service $\times$ format interactions were nontrivial: Gemini generally scored higher SUS than ChatGPT overall, while ChatGPT scored higher UMUX; however, ChatGPT exhibited a larger share of catastrophic (severity 4) issues. We discuss implications for prompt format selection and highlight recurrent failure modes (help & documentation, error prevention, consistency) for teams.

Keywords: Agentic AI, Prompt design, Usability guidance, SUS, UMUX, Heuristic evaluation, First pass adequacy

INTRODUCTION

Traditional human factors and UI/UX practice has matured within a systems engineering frame that emphasizes traceable requirements, task analysis, and iterative validation in context. Practitioners begin with user research to define operational scenarios, then allocate functions between humans and technology to optimize performance, safety, and life-cycle cost. Heuristics, standards, and accessibility guidance provide shared languages that connect design intent to testable interface qualities. Methods like heuristic inspection, cognitive walkthroughs, and benchmark usability testing are coordinated with engineering verification and validation plans. Instruments such as the System Usability Score (SUS) and workload measures complement effectiveness and efficiency metrics gathered during scenario-based trials.

Human-centered design ensures these considerations persist from concept through sustainment rather than appearing as a late-stage gate. Design artifacts are expected to be reusable across teams and phases, which encourages disciplined structure and explicit acceptance criteria. The result is a predictable workflow in which usability guidance is documented, versioned, and verified alongside code and configuration. This tradition has produced substantial gains in reliability and cost avoidance across diverse domains.

Agentic AI (AAI) systems built on large language models (LLMs) introduce a new production mode in which natural-language briefs can yield runnable code and interface scaffolds with minimal handoff friction. These systems promise to compress iteration loops by translating high-level intent into first-pass prototypes that are immediately testable. They also shift a portion of design representation from static artifacts to prompts that must be parsed, reasoned over, and enacted by stochastic models. While early successes are compelling, variability across services, prompt formats, and update cycles means the impact on established human-centered workflows remains unproven. What you ask for matters, but how you format guidance may determine whether a first pass is usable or brittle. In this sense, prompt design becomes "input UX," where constraint clarity, ordering, and example density are potential levers on quality. This paper describes initial explorations into framing this problem as a question of formatting rather than principle selection, pointing to the need for controlled comparisons across services. For example, compact, chunked, example-rich inputs might travel better across tools than narrative prose alone.

This initial study sought to isolate guidance format as the independent variable while holding constant the functional brief, evaluation tasks, and an expert rater panel. We compare two widely used AAI services executing the same "digital yard sale" (akin to a Craigslist or eBay resale e-commerce site) front-end build under four formats that vary in structure, constraint explicitness, and exemplar density. The design treats prompts as first-class specifications, with a control variant representing common practice and a meta-spec variant representing an AI-optimized pattern. Expert raters, blinded to service and format, complete a fixed task battery and score perceived usability and heuristic issues to reflect both user-centric and inspector-centric perspectives. Our analysis emphasizes descriptive contrasts and recurring failure modes that have direct remediation value for teams. By focusing on formatting, we avoid confounding due to shifting design principles and keep attention on the mechanics of communicating intent to AAI systems. The approach also acknowledges that organizations will mix traditional HFE artifacts with AI prompting rather than replace them outright. Findings are framed to inform prompt authoring practice, early-pass QA checklists, and lightweight governance for AAI-assisted delivery. In short, the work asks whether small, disciplined changes in how guidance is packaged can reduce rework and improve first-pass usability across tools.

METHOD

Study Design

We conducted a 2 (AAI Service) × 4 (Usability Guidance Format) comparative study that isolates guidance formatting as the independent variable while holding constant the functional brief, evaluation tasks, and expert rater panel. The two AAI services were OpenAI's GPT5 model and Google's Gemini 2.5 model. The four guidance variants (explained in more detail below) were: V0 Control (simply the instructions to "make it usable"), V1 Narrative-Minimal, V2 Checklist-Constrained, and V3 Meta-Spec (AI-ready). Each service generated one front-end prototype per variant (8 total prototypes). Each prototype was evaluated by three blinded expert raters using a fixed task script and standardized instruments (SUS, UMUX) plus a heuristic issue log with severity coding.

Expert Raters

Three HF/UX professionals with different backgrounds (a UX designer, a Human Factors Research Scientist, and a UI/UX researcher) from within our organization served as expert evaluators to rate the AAI prototype outputs. Raters were blinded to the guidance variant and the actual functional brief provided to the AAI to produce the prototype outputs, but were aware of what AAI used to generate each variant they assessed. No end users were recruited; this round focuses on first-pass expert inspection and perceived usability to characterize directional effects of prompt formatting.

Materials

Functional Brief (Constant Across Conditions)

All builds implemented a front-end-only "digital yard sale" prototype (akin to a simplified resale marketplace) with the following characteristics:

- Seeded catalog (≥8 items) and search/filter.
- Item detail pages (images, price, description, time listed).
- Question to seller interaction.
- Add to cart and mocked checkout (no real payments).
- CRUD for user-owned items.
- Output constrained to index.html, style.css, script.js, and README.md. No back-end code, authentication, analytics, or routing guards.

Guidance Variants (Independent Variable)

- **V0 Control:** Minimal instruction ("make it usable"); no structure.
- **V1 Narrative-Minimal:** Short, intent-oriented prose with light headings.
- **V2 Checklist-Constrained:** Hierarchical MUST/SHOULD/SHOULD NOT bullets; concise examples.
- **V3 Meta-Spec (AI-ready):** Fixed sections (**Purpose, Assumptions, Constraints, Acceptance-style criteria, Examples, Failure Modes**), with worked examples and check IDs.

Table 1: Usability guidance variant characteristics.

Key Dimension	Variant 0 – Control Condition	Variant 1 — Narrative-Minimal	Variant 2 — Checklist-Constrained	Variant 3 — Meta-Spec (AI-Ready)
Structure	None – simply "Ensure the application is usable"	Light structure; short narrative paragraphs with a few headings. Minimal hierarchy; intent-focused.	Highly structured bullets with clear hierarchy; sections per feature; MUST / SHOULD lists.	Fixed sections (Purpose, Assumptions, Constraints, Acceptance Checks, Examples, Failure Modes); strong hierarchy optimized for parsing.
Constraint Explicitness		Low–Moderate. Constraints mostly implied as good practice; few "must" statements.	High. Explicit MUST / SHOULD / SHOULD NOT requirements; unambiguous directives.	Very High. Testable acceptance checks with precise states/roles; constraints phrased as pass/fail criteria.
Exemplar Density		Very Low. 0–1 example (tone-setting only).	Medium. 1–2 concise pattern examples (e.g., sortable header, error summary).	High. 2–3 worked examples (markup/code skeletons) tied directly to the acceptance checks.
Domain Focus		Generic/Universal usability intent; applicable across tools and domains.	Generic/Universal but oriented to UI tasks; broadly portable across platforms.	Cross-tool, UI-engineering-oriented ("AI-ready") with WCAG/ARIA specifics; domain-agnostic but code-adjacent for front-end builds.

AAI Services

- **OpenAI GPT-5.0 via ChatGPT Pro interface**
- **Google Gemini 2.5 via Google Jules coding interface.**

Prompting Protocol

For each service × variant cell:

1. Provide the functional brief + one guidance variant verbatim.
2. Allow up to three clarification turns with the model to reach the baseline feature set (no researcher code edits).
3. Export and freeze the output as a ZIP containing index.html, style.css, script.js, and README.md.

Tasks and Evaluation Procedure

Raters executed the following six tasks for each prototype, recording observations and timing (optional):

1. **Post a new item** (complete a form with required fields and validation).
2. **Browse and search** the marketplace (filtering, result relevance, empty states).
3. **Open item details** (verify key attributes and image handling).
4. **Ask the seller a question** (compose, submit, and confirm).
5. **Add to cart and complete mocked checkout** (status visibility, error prevention, confirmation).
6. **Edit and delete** a user-owned item (feedback, undo/cancel behavior).

After completing the task battery for a prototype, raters submitted:

- **SUS** (10-item; 0–100 scoring).
- **UMUX** (4-item; 0–100 scoring).
- A **heuristic issue log** (Nielsen–Molich H1–H10 classification) with:
  - Issue description and evidence.
  - **Severity** rating (0=Not a problem, 1=Cosmetic, 2=Minor, 3=Major, 4=Catastrophic).
  - Optional weights for **Impact**, **Frequency**, **Persistence**, and a computed **Priority** score.

Measures

Primary Perceived-Usability Measures

- **SUS (0–100):** Standard scoring from the 10-item instrument.
- **UMUX (0–100):** Reverse-coded items applied per instrument guidance; mean rescaled to 0–100.

Heuristic-Inspection Measures

- **Issue count** per prototype.
- **Severity distribution** (0–4) and **actionable issues** defined as severity ≥ 2 (Minor/Major/Catastrophic).
- **Priority** summary (sum or mean of per-issue priority scores when present).

Derived/Quality Measures

- **Within-cell variability** (SD and range across 3 raters for each Service$\times$Variant).
- **SUS–UMUX correlation** at the Eval_ID level.
- **Top failure themes**: frequency of actionable issues by heuristic category (H1–H10).

Data Handling and Quality Control

- **Blinding:** All prototype bundles were anonymized; raters were not told variant.
- **Counterbalancing:** Prototype evaluation order was counterbalanced using a Latin-square-style sequence to mitigate fatigue/order effects.
- **Outliers:** We report cell-level means, SD, and ranges; given small n per cell (≈ 3), outliers are described qualitatively rather than trimmed.

Ethical Considerations

No human subjects or personally identifiable information were collected beyond professional participation of expert raters; end-user testing was not performed in this round. The study focuses on evaluation of software artifacts generated by AAI services under controlled prompting conditions.

RESULTS

Across 24 expert evaluations (3 raters × 2 services × 4 variants), aggregate SUS averaged 63.85 (SD = 21.91; N = 24) and UMUX averaged 51.52 (SD = 21.73; N = 22; one UMUX missing for Eval_ID 22). SUS and UMUX showed a moderate positive correlation ($r = 0.50$, $n = 23$ paired cases), indicating they trend together but are not redundant. Heuristic inspection yielded 240 logged issues in total, of which 107 were actionable (severity ≥ 2) and 24 were catastrophic (severity = 4). Within each Service×Variant cell ($\approx$3 ratings), rater-to-rater variability was notable (mean within-cell SUS SD $\approx$ 18.7, UMUX SD $\approx$ 22.6), so the analyses emphasize directional patterns and recurring themes rather than fine-grained inferentials.

Service patterns (pooled across variants). On SUS, Gemini scored higher than ChatGPT (67.69 vs. 59.32). On UMUX, ChatGPT scored higher than Gemini (54.58 vs. 48.96). Actionable-issue counts were similar (Gemini 55, ChatGPT 52), but ChatGPT's issue profile skewed more severe, with more catastrophic events (19 vs. 5) and a higher total priority burden, suggesting that when ChatGPT fails it tends to fail "harder."

Variant patterns (pooled across services). V0 (Control) delivered the strongest perceived usability and the lightest issue burden: SUS = 77.08, UMUX = 63.33, 19 actionable issues, and 3 catastrophic (lowest). V1 (Narrative-Minimal) and V2 (Checklist-Constrained) landed mid-pack on perceived usability (SUS $\approx$ 66 and 55; UMUX $\approx$ 52–53) with higher actionable-issue totals (V1 = 28; V2 = 31). V3 (Meta-Spec/AI-ready) showed lower perceived usability (UMUX = 39.58) and the highest catastrophic count (8). In this dataset, additional formatting structure did not outperform the minimal Control; more structure correlated with lower perceived usability and heavier issue profiles.

Service × Variant interaction. V0 (Control) was consistently strong for both services (SUS: ChatGPT = 81.25, Gemini = 75.00; UMUX: ChatGPT = 72.92, Gemini = 56.94). For V1/V2, Gemini exceeded ChatGPT on SUS (V1: 73.33 vs. 58.33; V2: 71.67 vs. 38.33), while UMUX results were mixed (ChatGPT $\geq$ Gemini on V1; Gemini $\geq$ ChatGPT on V2). For V3, ChatGPT showed higher SUS than Gemini (66.67 vs. 48.33), but ChatGPT+V3 concentrated the most catastrophic issues (n = 7), indicating risk for that pairing despite acceptable SUS. Overall, the data suggest that prompt-format "fit" is service-specific; teams should pair format with service rather than assume a universal "structured-wins" rule.

Stability, missingness, and noise considerations. Within-cell variance was large enough to caution against small-effect claims; however, rank-order patterns were stable: V0 generally led on both SUS and UMUX and carried

the fewest serious issues. A single missing UMUX (Eval_ID 22) did not alter the substantive conclusions under available-case summaries.

IMPLICATIONS

The findings suggest teams should treat prompt formatting as a tunable control surface, not a doctrine. In our data, a minimal control prompt outperformed heavier formats on first-pass usability and carried fewer severe issues, indicating that over-structuring can inadvertently divert model effort from basics like status visibility, help text, and error prevention. Practically, this argues for service-specific pilot runs before standardizing on any "AI-ready" spec: A/B the control against your preferred structured pattern on the exact service you'll use, then lock the winner into your delivery playbook. Treat prompts as first-class specifications, but resist the impulse to enumerate every constraint at once; start light, measure, and only add structure that demonstrably improves first-pass outcomes for that service.

From a workflow perspective, organizations should integrate AAI into the existing HFE toolchain with lightweight governance. Establish a small prompt library (control + one structured variant) per service, a first-pass QA checklist aligned to the top failure modes, and automated conformance checks where possible (contrast, labels, keyboard reachability). Track a few operational KPIs, e.g., SUS/UMUX at first pass, severity-4 count, and hours to test-ready, to decide when a structured spec is earning its keep. Be mindful of catastrophic-issue clustering in certain pairings (e.g., ChatGPT with the heaviest spec in our study) and gate those paths behind extra review until validated. Finally, keep the human-factors loop in place: use experts to spot systematic gaps, harvest fixes back into prompts and components, and evolve the prompt patterns the same way you evolve your design system.

CONCLUSION

In this initial comparative study, we treated prompt formatting as a controllable input to Agentic-AI builds and found that a minimal control prompt produced the strongest first-pass usability with the fewest severe issues, while heavier, highly structured formats did not generalize their presumed advantages across services. Service×format interactions mattered: Gemini tended to score higher on SUS, ChatGPT on UMUX, yet ChatGPT concentrated more catastrophic issues in certain structured pairings, underscoring that "best" formatting is service-specific rather than universal. Recurrent failure modes (e.g., help/onboarding, error prevention, and, notably, consistency) appeared across all conditions, offering clear, high-leverage targets for early remediation regardless of tool or format. Practically, teams should pilot and tune prompt formats per service, keep a lightweight control baseline in the library, and instrument first-pass KPIs (SUS/UMUX, severity-4 counts, hours to test-ready) to decide when added structure earns its keep. These results argue for integrating AAI within existing HFE governance (treating prompts as first-class specifications) while expanding QA to include small, empirical A/Bs that verify format–service fit before scaling.

Appendix: Functional Specification/Application Prompt

This appendix provides the verbatim functional specification that was input as a prompt to the AAI services.

GOAL:

Design and develop a front-end prototype "digital yard sale" web application that allows a user to post and manage items they own for sale, browse a marketplace of items posted by other users, and purchase items they want from the marketplace via a direct point of sale purchase system. This prototype should feature full front-end functionality to permit usability testing of the front end workflows and features. This prototype does not need to include backend functionality or integrations with third-party services (e.g., payment services for the point of sale system).

Instructions:

I1. The application must be fully self-contained and not require any significant backend. It should specifically be composed of only the following files: index.html, script.js, style.css, and README.md

I2. The application's design and features must adhere to the [USABILITY GUIDANCE] provided below to ensure a positive user experience.

I3. The app must run locally in a browser without any backend.

I4. Where necessary, encode a static set of mock data to enable all front-end features to function. For example, create a set of new mock listings a user to can add as if they just filled out a new item listing form. For example, create a set of existing mock listings so the marketplace is populated with content on initial load.

I5. Out of scope application elements include: Authentication, routing guards, payments, analytics, server code.

I6. Generate a readme file to accompany the application that explains: (a) each feature / functionality of the application; and (2) how the design of the application adheres to each of the [USABILITY GUIDANCE] elements.

Strict Requirements

R1. The application must provide a default marketplace of 8 different mock items upon loading. Each item should have a mock image, title, and price. Each item should also have a description and time listed / how long its been on the site, that can be viewed by selecting a given item for details.

R2. The application must permit the user to execute a mock purchase / checkout process for one or more marketplace items.

R3. The application must permit the user to create, read, update, and delete their own mock items that then appear in the marketplace.

R4. The application must permit the user to search for items of interest.

R5. The application must permit the user to ask the seller a question.

Remember:

The goal of this application is to permit usability testing on the front-end interface, so the front-end design, functionality and workflows are a priority.

Result:

Your final output should be a self-contained application that can be run in any browser offering full-front-end functionality and any required mock data to enable that functionality for usability testing.

[USABILITY GUIDANCE]
//insert variant specific usability guidance here//
[End of USABILTY GUIDANCE]

REFERENCES

Brooke, J. (1996) 'SUS: A "quick and dirty" usability scale', in P. W. Jordan, B. Thomas, B. A. Weerdmeester and A. L. McClelland (eds.), *Usability Evaluation in Industry*. London: Taylor & Francis, pp. 189–194.

Finstad, K. (2010) 'The Usability Metric for User Experience (UMUX): A simple and reliable construct of perceived usability', *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 54(1), pp. 229–233.

Nielsen Norman Group (2020) '10 Usability Heuristics for User Interface Design'. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/.