From Feelings to Metrics: Understanding and Formalizing How Users VIBE-TEST LLMs

Itay Itzhak
Technion – Israel Institute of Technology
The Hebrew University of Jerusalem
itay1itzhak@gmail.com

Eliya Habba
The Hebrew University of Jerusalem
eliya.habba@mail.huji.ac.il

Gabriel Stanovsky
The Hebrew University of Jerusalem
gabriel.stanovsky@mail.huji.ac.il

Yonatan Belinkov
Technion – Israel Institute of Technology
belinkov@technion.ac.il

Abstract

Evaluating LLMs is challenging, as benchmark scores often fail to capture models' real-world usefulness. Instead, users often rely on "vibe-testing": informal experience-based evaluation, such as comparing models on coding tasks related to their own workflow. While prevalent, vibe-testing is often too ad hoc and unstructured to analyze or reproduce at scale. In this work, we study how vibe-testing works in practice and then formalize it to support systematic analysis. We first analyze two empirical resources: (1) a survey of user evaluation practices, and (2) a collection of in-the-wild model comparison reports from blogs and social media. Based on these resources, we formalize vibe-testing as a two-part process: users personalize both *what* they test and *how* they judge responses. We then introduce a proof-of-concept evaluation pipeline that follows this formulation by generating personalized prompts and comparing model outputs using user-aware subjective criteria. In experiments on single-turn coding benchmarks, we find that combining personalized prompts and user-aware evaluation can change which model is preferred, reflecting the role of vibe-testing in practice. These findings suggest that formalized vibe-testing can serve as a useful approach for bridging benchmark scores and real-world experience.[1]

1 Introduction

Evaluating LLMs has been a long-standing challenge in NLP research (Laskar et al., 2024; Cao et al., 2025), as popular evaluation suites typically report performance as aggregated scores on standardized tasks (Zhang et al., 2024; Jain et al., 2025). However, these scores often miss the usefulness of models in real-world workflows (Kiela et al., 2021; Mazumder et al., 2023; OpenAI, 2025a). In practice, model usefulness often depends on context-dependent criteria, such as clarity, ease of use, or workflow fit (Weidinger et al., 2025; Saad-Falcon et al., 2024). As a result, strong benchmark scores do not necessarily imply a good fit for users' needs in everyday tasks.

In response, many users turn to "*vibe-testing*": an informal practice of evaluating models through targeted experiments or extended personal use (Davies, 2025; huggingFace, 2025). Instead of relying solely on benchmark scores, users compare models on tasks that resemble their own workflows and judge the responses qualitatively. These comparisons often focus on practical aspects of model behavior, such as writing style, clarity of explanations, ease of use, or how well the output fits a specific workflow (Figure 1). Vibe-testing can be performed by individual users or shared by community members on blogs, forums, and social media.

[1]See code and study artifacts at: https://technion-cs-nlp.github.io/vibe-testing-llms.

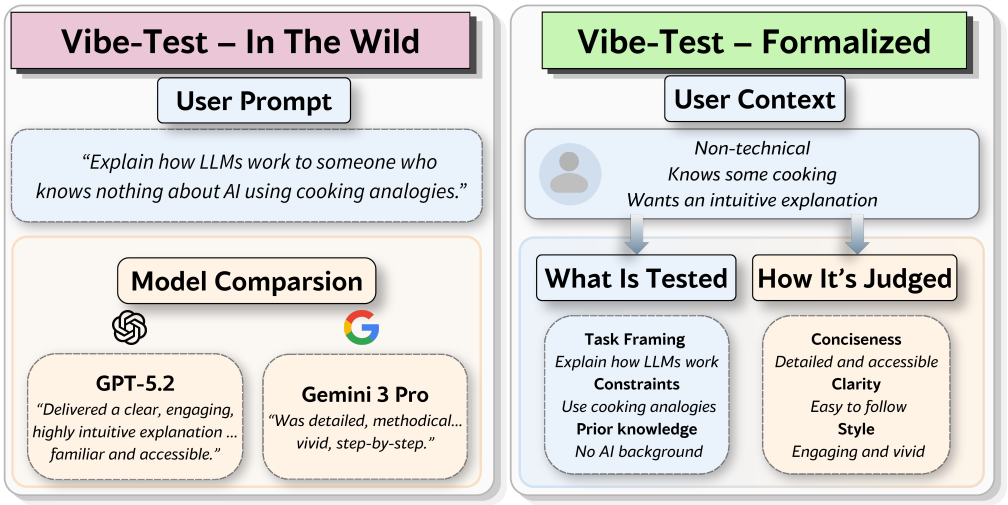


Figure 1: **Anatomy of a "vibe-test".** In practice, users evaluate LLMs by "vibe-testing" them: – writing personalized prompts that test specific behaviors and judging models' responses using personal subjective criteria. We analyze recurring patterns of vibe-testing in real-world user comparisons, formalize them into a two-part structure, and present a proof-of-concept pipeline for automated vibe-testing. Example taken from Tom's Guide.

Users' reliance on vibe-testing implies that it captures valuable aspects of model performance that benchmarks often miss. However, vibe-testing is inherently informal and subjective – different users test different tasks and judge responses from their own perspective. As a result, insights from these evaluations remain scattered and fragmented, making them difficult to compare or transfer across settings. This informality leaves a gap between the practical insights of vibe-testing and our ability to study them systematically. Recent work has suggested assessing model "vibe" and personalized evaluation, but none have empirically studied vibe testing itself as a user practice or suggested an evaluation framework grounded in it (Section 2).

In this work, we empirically study the practice of vibe-testing, formalize its distinctive patterns, and propose a proof-of-concept evaluation pipeline for systematic analysis. We begin by examining vibe testing in practice, drawing on two empirical sources. First, we conduct a survey on evaluation practices, asking questions such as *"What do you look for when testing a model?"*. Second, we collect an "in-the-wild" corpus of model comparisons from blogs, forums, tech articles, and YouTube reviews. We annotate these examples to identify recurring patterns in users' test design and response evaluation. Together, these sources provide real-world evidence of how users vibe-test models: what they test and what they look for when judging responses.

Building on these empirical sources, we formalize vibe-testing as an evaluation practice defined by two recurring types of dimensions. *Input dimensions* capture what users test and how they construct prompts, while *Output dimensions* capture how users judge model responses. For example, in coding assistance, input dimensions can include the type of coding task or the amount of context provided (e.g., debugging a codebase). Output dimensions can include clarity, adherence to constraints, and fit to the user's workflow (e.g., production-ready code). This formalization makes vibe-testing easier to compare and analyze, and provides a basis for systematic reproduction.

We leverage this formalization and introduce a proof-of-concept evaluation pipeline that mirrors the two-part structure of vibe-testing. Given a brief user description, the pipeline first rewrites benchmark prompts to reflect that user's likely context and preferences. It then compares models head-to-head by judging their responses along user-relevant output dimensions from that user's perspective. We apply the pipeline to single-turn coding

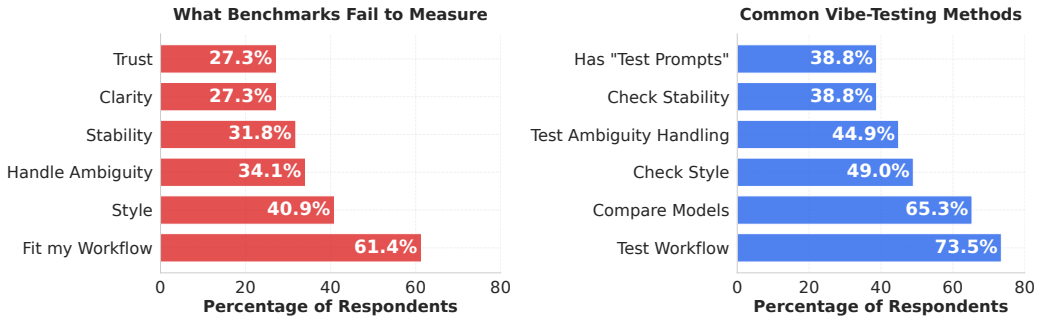


Figure 2: **Benchmarks vs. vibe-testing in practice. Left: What benchmarks miss.** Survey participants selected real-world qualities that benchmarks fail to capture (multi-select), including workflow and style fit, handling ambiguity, stability, clarity, and trust. **Right: How users test models.** Common strategies include trying tasks from one's own workflow, side-by-side comparisons, probing style, stress-testing ambiguity handling, checking repeated runs stability, and using recurring "test prompts,". Notably, the qualities participants say benchmarks miss largely match what they explicitly probe during vibe-testing.

benchmarks and find that personalizing both the prompt and the judgment criteria can change which model is preferred. In several head-to-head comparisons, the preferred model flips relative to the original benchmark prompts, while non-personal rewrites largely preserve the original ordering. These results echo the core idea behind vibe-testing: model preferences can change when both the task framing and response judgment are tailored to the user.

Overall, this work takes a first step toward turning vibe-testing from an informal practice into a structured form of user-centered evaluation.

2 Background: personal vibe evaluation

Recent work has shown that standard benchmarks often miss aspects of model performance that matter to users in real-world use, motivating more qualitative user-centered evaluation (Cao et al., 2025; Weidinger et al., 2025). However, systematically capturing vibe-testing evaluation requires a framework that is subjective, personalized, and grounded in empirical evidence. To our knowledge, prior work typically captures only one of these properties.

Vibe-based subjective evaluation. A few recent papers study "vibe"-like aspects of model behavior, but they do so in different ways. Most relevant is VIBECHECK (Dunlap et al., 2024), which measures qualitative "vibe" differences between models, but at the population level rather than for individual users. VIBE CHECKER (Zhong et al., 2025) adds verifiable instruction checks to coding tasks, but covers only automatically checkable traits and leaves out softer subjective dimensions, limiting potential coverage. Beyond these, HELM Instruct (Zhang et al., 2024) offers general stylistic evaluation while ChatBench (Chang et al., 2025) focuses on evaluating interactive conversations, both relying on predefined criteria.

User-focused evaluation. EvalLM (Kim et al., 2024) supports manual rubric customization but does not automate or infer personalization. IQA-Eval (Li et al., 2024) adapts evaluation to user personas by simulating interactive correction and questioning, but focuses on the writing style of factual questions. EVALAGENT (Wadhwa et al., 2025) mines expert-authored guidance to uncover implicit evaluation criteria, but targets prompt underspecification rather than personalizing the user's input. Complementary work uses LLM-based user simulators for interactive evaluation in task-oriented dialogue (Luo et al., 2024; Jia et al., 2024), but does not address subjective model comparison across user profiles.

Our work addresses the qualitative user-centered evaluation gap in a different way than previous work. We aim to capture vibe-testing evaluation by empirically examining how

users compare models in the real world, formalizing this process, and proposing a modular pipeline that can be extended with other methods.

3 What is vibe-testing?

To study vibe-testing systematically, we first ask what it looks like in practice. To answer this, we collect and analyze two complementary empirical resources: a survey of user evaluation practices and an analysis of in-the-wild comparison reports. Together, these sources provide a clearer and more concrete picture of how users evaluate models in practice.

3.1 User survey

We conduct a survey to understand the prevalence of vibe-testing, how users carry it out in practice, what they think benchmarks miss, what vibe-testing helps them assess instead, and whether it's worth automating (full survey results are in Appendix A).

Demographics. We recruit 51 volunteers via social media platforms (e.g., X/Reddit), including both AI/ML experts (47%) and broader technical practitioners (47%), as well as non-technical users (6%). Respondents reported using AI tools daily (92%) or weekly (8%).

Prevalence of vibe-testing. We ask respondents whether they have vibe-tested models, loosely describing vibe-testing as *"Evaluating an AI model through direct interaction, using your own prompts or tasks to judge how the model performs in practice."* Most respondents reported "Yes" (82%) and that they often experiment with models (mean=5.31 on a scale of 1–7).

How users vibe-test models. We ask respondents how they test models and how they judge models' outputs. The most common testing methods were trying tasks from one's own workflow and comparing models' outputs side by side (Figure 2, right). When asked which criteria they use to judge outputs, the most frequently selected were correctness (92%), clarity (59%), and workflow fit (41%). These responses suggest that vibe-testing is typically grounded in personal workflows and judged using both correctness and practical, user-relevant criteria.

The benchmark-experience gap. We then ask respondents whether they had ever encountered a model that "felt" significantly different from what its benchmark scores would suggest. Most answered "Yes" (86%), indicating that many perceive a mismatch between benchmark rankings and real-world experience. When asked what benchmarks fail to measure, workflow and style fit were the most common selections (Figure 2, left). Notably, these reported gaps align closely with the aspects respondents say they evaluate when vibe-testing, as illustrated side by side in Figure 2. Finally, most respondents (83%) expressed interest in tools that could make the vibe-testing process more structured or automated.

The survey findings suggest that vibe-testing is common among technical users and is built around personal workflow tasks, side-by-side comparisons, and subjective judgments of output. They also point to a perceived gap between benchmarks and real-world experience, and to the value that vibe-testing adds. We next complement these findings by analyzing public in-the-wild model comparison reports to examine recurring patterns in real-world vibe-testing.

3.2 Analyzing vibe-testing in the wild

We next turn to "in the wild" model-comparison reports to examine how vibe-testing appears in practice. Unlike benchmark results, these comparisons are typically shared informally across social media, blogs, and community forums.

We semi-automatically construct a carefully curated corpus of 40 public model comparison reports in four stages:[2]

[2]Additional details on the corpus are in Appendix B.

(1) **Source collection.** We manually search for public reports that contain concrete, qualitative comparisons of LLMs, drawing from dozens of YouTube reviews, Reddit threads, blog posts, and news articles (source list is in Appendix B). We include sources that (i) reference specific models, (ii) describe at least one concrete test input (prompt, task, or scenario), and (iii) include qualitative judgments and subjective claims. The resulting corpus is a selected collection of naturally occurring vibe-testing examples.

(2) **Vibe-test instances extraction.** For each comparison report, we use LLMs[3] to extract and label *vibe-test instances*, defined as localized cases where a report author evaluates one or more models on a specific input using some qualitative criteria. Specifically, the LLMs were prompted to return short quoted spans or paraphrased snippets corresponding to vibe-tests, along with the tested task and the stated criteria. We manually verified extracted *vibe-test instances*, removing false positives and correcting errors.

(3) **Attribute annotation.** We annotate each *vibe-test instance* with a small set of structured attributes, including task type, models compared, and the subjective criteria mentioned in the source (e.g., "Answer Clarity"). We perform this annotation with LLM assistance and manual review and refinement to ensure consistency and faithfulness to the original text.

(4) **Consolidating dimensions.** To characterize recurring patterns, we ask the LLMs to propose lists of repeated subjective dimensions appearing across *vibe-test instances* by grouping similar criteria under shared labels (e.g., "Answer Clarity" and "Clear Output" under "Clarity"). In parallel, we independently compiled our own lists from manual reviews. We then iteratively reconcile and refine these lists, using both our judgments and LLM suggestions, to derive a final set of recurring dimensions. Using this fixed set, we re-annotate *vibe-test instances*, establishing the final dimension labels for the corpus.

The iterative procedure in Stage (4) yields a consolidated list of recurring dimensions spanning both the *input*, what users choose to test and how they frame it, and the *output*, what qualities they attend to when interpreting responses. Together with our survey results, this list provides the basis for the formalization of vibe-testing presented next.

4 Formalizing vibe-testing

Based on our empirical findings in Section 3, we now formalize a definition of vibe-testing.

Definition 1 (Vibe Testing) *Vibe-testing is an interaction-based LLM evaluation practice, in which evaluators adapt both the* ***input*** *they test (via input dimensions) and the* ***criteria*** *used to judge output (via output dimensions). It is intended to capture aspects of practical utility and user experience that standard benchmarks may miss.*

Dimensions of Vibe-Testing. To describe vibe-testing systematically, we introduce vibe dimensions: recurring aspects of what users test and how they judge outputs. Our survey and corpus suggest two broad groups of such dimensions.[4]

- **Input-oriented dimensions** consist of *task type*, *task complexity/scope*, *real-world context setting*, *persona-based framing*, *underspecification level*, *constraint tightness*, and *reference material availability*.
- **Output-oriented dimensions** consist of *comparison setup*, *correctness/accuracy*, *clarity and structure*, *cognitive load*, *style/tone fit*, *workflow fit*, *friction/loss of control*, *ambiguity handling*, *reliability/stability*, *trustworthiness/safety behavior*, and *anthropomorphism*.

We illustrate these dimensions with the vibe-test example from Figure 1. On the input side, the prompt reflects choices about *real-world context setting* (cooking) and *persona-based framing* (a non-technical audience), and it sets a moderate *task complexity/scope* (a conceptual explanation rather than a single fact). An evaluator can further specialize the same test with

[3]GPT-5.2 and Gemini 3 Pro were chosen for this task via "vibe-testing". Prompts are in Appendix B.
[4]Tables 2 and 3 in the Appendix list the definitions and illustrative cues for each dimension.

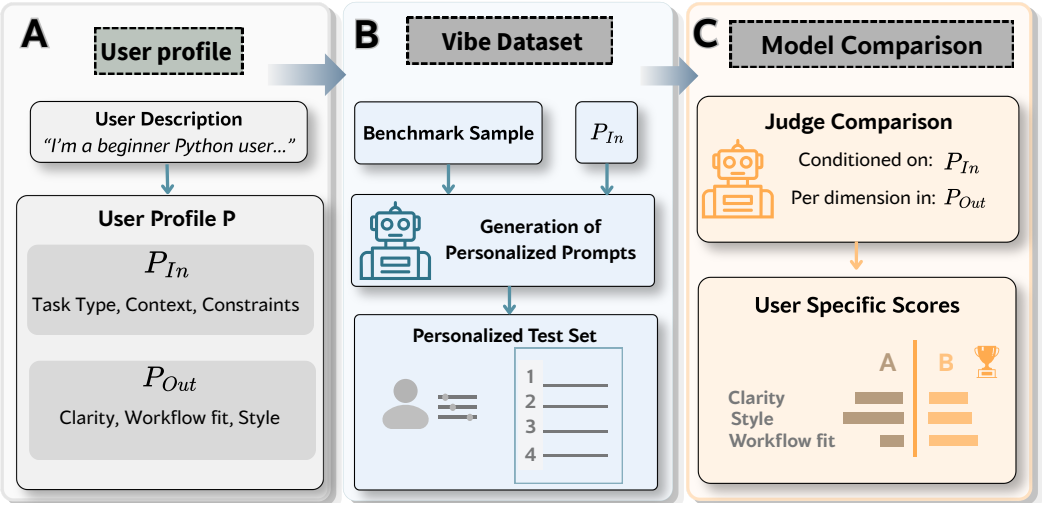


Figure 3: **Automatic Vibe-Testing Pipeline**: Given a user description, the pipeline **(A)** constructs a user profile P (composed of input P_{in} and output P_{out}) preferences, **(B)** rewrites benchmark samples into a personalized prompt aligned with P_{in}, and **(C)** compares responses using P_{out} to produce per-dimension head-to-head model comparisons.

constraint tightness (e.g., "use exactly three analogies") or by increasing *underspecification level* (e.g., leaving the target audience implicit).

On the **output side**, the example setting shows a common *comparison setup* in which the same prompt is run on two models and the responses are read side by side. The response judgment reflects multiple dimensions: The mention of *"clear...highly intuitive"* points to the *clarity and structure* dimension, while *"detailed, methodical"* indicates their preferred *tone/style* fit, and *"familiar and accessible"* refers to the evaluator's own *workflow fit*. Different evaluators can run the same prompt with similar judgment criteria in mind, but interpret those criteria differently or assign them different importance, leading to different preferences even when both responses are correct. In the next Section, we present an evaluation pipeline that reflects this formulation of vibe-testing – adapting both the input and the evaluation based on user preferences.

5 Automating vibe-testing

We now instantiate the formulation above as a modular proof-of-concept pipeline (Figure 3). Its goal is to test whether such a two-part evaluation method can reveal meaningful preference shifts that benchmarks miss. We study it in a focused setting using single-turn coding tasks and pairwise comparisons as a concrete testbed. The resulting pipeline is modular: its stages can be implemented in different ways and expanded upon, using existing or new methods for user profiling, input personalization, and subjective evaluation.

5.1 Pipeline description

Given a brief user description, the pipeline first builds a structured profile of the user's input and output preferences. It then uses that profile to rewrite personalized benchmark prompts and compare candidate models head-to-head from the same user perspective. The result is a set of user-conditioned comparisons that lets us quantify how model preference changes across users and prompt variants.

(A) User profiling. We begin by converting a natural language user description (e.g., "I'm a novice Python student") into a structured user profile $\mathcal{P}$ using an LLM. The profile

includes the user's preferred input dimensions ($\mathcal{P}_{in}$) and output dimensions ($\mathcal{P}_{out}$). This structured profile is then used to guide both prompt personalization and output evaluation.

(B) Vibe dataset construction. To personalize what is being tested, we rewrite benchmark prompts based on the user's input dimensions. For each benchmark sample s and profile $\mathcal{P}$, we generate K variations of the original prompt. We do this by first generating a small set of editing options according to $\mathcal{P}_{in}$, such as "request concise answer" or "emphasize efficiency". To create a new prompt variant, we sample a combination of these options and apply them to the original prompt. We then run a semantic-preservation verification using an LLM to flag variants that are likely to change the task intent (after pipeline refinement, almost all variants pass). The resulting "vibe dataset" pairs each canonical benchmark sample s with a set of K controlled prompt variants conditioned on $\mathcal{P}_{in}$.

(C) Model Comparison. We evaluate both whether a model solves the task and how well its response fits the user's preferences. To measure correctness, we compute Pass@1 on the benchmark tests. To measure response preference, an LLM judge compares two model outputs side by side from the same user perspective. For each output dimension in $\mathcal{P}_{out}$, the judge chooses the preferred response for that user along with a confidence score and rationale. These pairwise judgments are aggregated into win rates to quantify shifts in model preferences across users.

The resulting pipeline is a personalized evaluation suite that reflects the user's input and output preferences. For each user, the pipeline yields a tailored set of test prompts and interpretable pairwise personal judgment scores. These judgments can be inspected at the dimension level, comparing models on specific dimensions across samples, or aggregated into an overall preference signal. In this way, the pipeline captures both which model is preferred and why. In the following section, we evaluate the pipeline on coding tasks across four profiles and four model matchups.

5.2 Practitioner's guide

Practitioners can use the pipeline for adapting evaluations to a target user or community. They can construct a user profile from direct descriptions, surveys, or other available user data, then use its input dimensions to personalize an existing benchmark or task collection and its output dimensions to define the evaluation criteria. Results can be analyzed by dimension or aggregated across users, and open-ended tasks can use human review or task-specific rubrics. See Appendix G for the full Practitioner's guide.

6 Experiments

6.1 Experimental setup

Models. We study four head-to-head matchups between related models: (1) GPT-5.1 (OpenAI, 2025b) vs. GPT-OSS-20B (Agarwal et al., 2025), (2) GPT-5.1 vs. GPT-4o (OpenAI, 2024), (3) Gemini-3 Pro (Pichai et al., 2025) vs. Gemma-3 4B (Kamath et al., 2025), and (4) Qwen3-32B vs. Qwen3-14B (Yang et al., 2025). These pairings allow us to test whether personalization reveals finer-grained trade-offs when the models being compared have an expected capability ordering (due to size differences or provider tiers). Prompt personalization is done using GPT-5.1 and Qwen3-32B, with $k = 2$ and 3 variations, respectively. Unless stated otherwise, we use GPT-5.1, GPT-OSS-20B, and Qwen3-14B as LLM judges (GPT-5.1 omitted for Gemini and Qwen comparisons due to cost). We report judge agreement percentages and Cohen's Kappa.[5]

Data and prompt variants. We use the MBPP+ and HumanEval+ Datasets (Liu et al., 2023), sampling 100 problems from each. For each persona and problem, we evaluate models under three prompt types: the original prompt, K personalized variants, and K

[5]See Appendix E for more implementation details.

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	Researcher	Advanced
GPT-5.1	Original	0.03* (0.00)	0.01* (0.00)	0.08* (0.02)	**0.91*** (0.01)
vs.	Personalized	**0.77*** (0.02)	**0.55*** (0.02)	0.43* (0.01)	**0.58*** (0.03)
GPT-OSS-20B	Control	0.06* (0.00)	0.02* (0.00)	0.10* (0.02)	**0.93*** (0.01)
GPT-5.1	Original	0.09* (0.00)	0.16* (0.02)	**0.63*** (0.02)	**0.88*** (0.00)
vs.	Personalized	**0.94*** (0.01)	**0.77*** (0.02)	**0.97*** (0.00)	**0.82*** (0.02)
GPT-4o	Control	0.08* (0.01)	0.19* (0.03)	**0.70*** (0.02)	**0.95*** (0.01)
Gemini-3-Pro	Original	0.48 (0.02)	0.53 (0.01)	0.48 (0.03)	**0.66*** (0.02)
vs.	Personalized	**0.93*** (0.01)	**0.90*** (0.01)	**0.97*** (0.01)	**0.70*** (0.02)
Gemma-3-4B	Control	0.44* (0.02)	0.45* (0.02)	0.48 (0.02)	**0.69*** (0.02)
Qwen3-32B	Original	0.54 (0.04)	**0.62*** (0.03)	0.53 (0.04)	0.38* (0.05)
vs.	Personalized	0.54 (0.02)	**0.67*** (0.04)	**0.62*** (0.03)	**0.54*** (0.04)
Qwen3-14B	Control	0.54 (0.05)	**0.59*** (0.04)	**0.55*** (0.06)	0.40* (0.05)

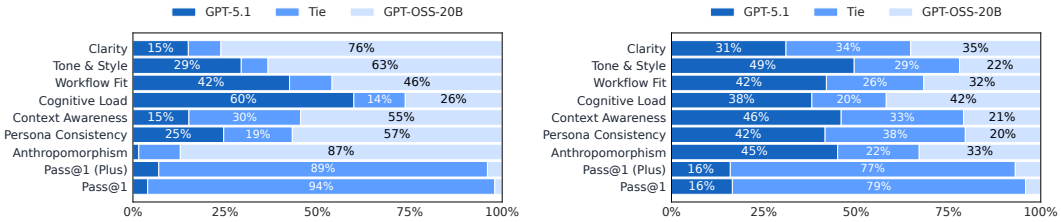
Table 1: **Personalization shifts win-rates.** Per-sample win rates on MBPP+ by user and prompt type, reported for the first model in each pair (tie-rate in parentheses). On benchmark prompts, GPT-5.1 underperforms for Beginner and Intermediate, but personalized rewrites sharply increase its win rate, as they do for Gemini-3-Pro. Both remain stronger for the Advanced persona in all prompts. For Qwen3-32B vs. Qwen3-14B, personalization yields a weaker shift toward the larger model. Control shows generic prompt rewrites match the original pattern. * denotes statistical significance on a two-sided binomial test.

neutral paraphrase controls produced with PROMPTSUITE (Habba et al., 2025). The control prompts change the wording without adding persona-specific information (e.g., adding "Perform the following task:"), which helps isolate the effect of personalized paraphrasing. We evaluate one generation per prompt due to cost, as partial experiments with GPT-5.1, GPT-OSS-20B, and Qwen models show a consistent win rate. To check task preservation after personalization, we evaluate the correctness of personalized prompts using the original benchmark tests and report the preservation rate: the percentage of samples solved with both the original prompt and personalized rewrite. This is only a lower bound, since failures may also result from additional constraints or changes in response format. We report this for GPT-5.1 and Gemini-3 Pro and manually inspect 20 prompts that fail this check.

Personas and vibe dimensions. We use four author-written controlled user personas representing varying levels of coding expertise archetypes: *Beginner Student*, *Intermediate Learner*, *AI Researcher*, and *Advanced Developer*. Each persona includes a description specifying both input and output preferences, and assigns importance weights from 1 to 5 to the output dimensions. We also evaluate two real-life personas derived from public user data as a limited check that the observed effects extend beyond the controlled personas (See Appendix D.1). To control for prompt length, we also test one- to three-word personalized variants for the Beginner and Advanced personas (See construction details and results in Appendix D.2).

Because our experiments focus on single-turn coding, we evaluate the dimensions most relevant to that setting, excluding *Stability* and *Safety*. We further replace *Friction* and *Ambiguity handling* with two narrower dimensions: *Context awareness*, which captures whether the response respects the task context and constraints, and *Persona consistency*, which captures whether the response fits the intended user role.

Pairwise judging and aggregation. For each persona, sample, and output dimension, an LLM judge assigns a pairwise label (A WINS/B WINS/TIE). To mitigate position effects, we evaluate each comparison twice with the responses order-swapped following Jiang et al. (2025b) and resolve disagreements using the judge's confidence score. To determine



(a) **Original prompts** (b) **Personalized**

Figure 4: **Personalization changes model preferences.** Head-to-head win rates for GPT-5.1 vs. GPT-OSS-20B on MBPP+, broken down by dimensions. **Left:** original benchmark prompts. **Right:** persona-specific rewrites averaged over four personas. Several dimensions favor different models depending on the prompt form, showing that benchmark prompts can mask user-relevant differences beyond correctness.

the per-sample winner, we first compare *Pass*@1 correctness: if only one model is correct, it wins. Otherwise, we aggregate the output dimensions judgments using the persona's weights. We report per-dimension and overall win and tie rates across all judges. [6]

Human validation of LLM judgments. We validate the automated judging setup with a human preference study. Six graduate student annotators compare model outputs in the same pairwise, persona-conditioned format used for LLM judges, selecting a winner or tie across dimensions. This task is demanding – it requires code understanding and careful comparison of lengthy responses across seven dimensions. We therefore keep the study small, with each annotator completing 12 easier-to-annotate comparisons in about one hour. The study covers two personas (Beginner, Advanced), two model pairs (GPT-5.1 vs. GPT-4o, Gemini-3-Pro vs. Gemma-3-4B), and 32 sampled comparisons (24 original, 8 personalized), filtered for length and a minimal 2-judge consensus. This checks whether our persona-conditioned LLM judgments align with human preferences (see Appendix F).

6.2 Results

Personalization changes which model is preferred. As shown in Table 1, persona-specific rewrites often reverse the pattern seen under the original benchmark prompts. For GPT-5.1 vs. GPT-OSS-20B and GPT-5.1 vs. GPT-4o, the original prompts favor the weaker or older model for the Beginner and Intermediate personas, whereas personalized prompts sharply increase GPT-5.1's win rate. For the more expert persona, GPT-5.1 is already mostly preferred on the original prompts and remains preferred after personalization. Gemini-3 Pro vs. Gemma-3-4B preference is balanced on original prompts and led by Gemini-3 Pro on personalized prompts, with a substantially weaker shift for Qwen3-32B vs. Qwen3-14B. In contrast, control paraphrases mostly preserve the original ordering, suggesting that the shift is unlikely to stem from generic rephrasing variations.

Dimension-level shifts . The dimension-level results help explain why overall preferences shift, as shown in Figure 4. On the original prompts, GPT-OSS-20B is often favored over GPT-5.1 on clarity and tone, while GPT-5.1 leads on cognitive load. After personalization, GPT-5.1 becomes more competitive across these subjective dimensions, reversing several gaps. Manual inspection suggests that GPT-5.1 often defaults to concise, solution-focused responses, whereas GPT-OSS-20B gives verbose, tutor-like answers. Responses to benchmark prompts may therefore mask user-relevant trade-offs by favoring one interaction style.

The effect generalizes across evaluation settings. We observe mostly similar trends when using Qwen3-32B as a generator, when evaluating on HumanEval+, and varying the winner rule, including unweighted scoring, majority vote, and strict tie-breaking (Appendix D).

[6]See Appendix C for experimental details, and Appendix D for robustness analysis and ablations

The task preservation check shows 87% $\pm$ 2%, indicating personalization mostly preserves the original task (20 failed checks were manually verified to also preserve the original task).

The preferences of LLM judges are consistent and align with human judgment. Across samples, LLM judges reach reasonably consistent per-sample preferences with a mean agreement of 78% $\pm$ 13 and Fleiss's $\kappa = 0.39 \pm 0.16$. Human validation on the original prompts shows high agreement both within humans (94% $\pm$ 15, $\kappa = 0.80 \pm 0.39$) and between humans and LLM judges (89% $\pm$ 16, $\kappa = 0.78 \pm 0.35$). Agreement is lower on the small personalized-prompt subset, where longer responses increased annotation difficulty. We therefore treat the original-prompt results as support for the judging protocol, while the personalized results remain inconclusive (Appendix D). Overall, these results support the reliability of preference judgments across users and samples.

7 Conclusion

We study vibe-testing as a real-world evaluation practice and present an empirically grounded formalization. We turn this formalization into a proof-of-concept pipeline and show that personalization can change model preferences. More broadly, our findings show that model quality depends not only on what a benchmark measures, but also on how users frame tasks and judge responses. Formalized vibe-testing provides a practical foundation for future evaluation methods that systematically model user-dependent preferences.

8 Limitations and future work

Our empirical grounding is limited by both sources: the in-the-wild analysis relies on LLM-assisted extraction and manual verification, which limits scale and remains subjective, and the survey includes 51 participants from a highly technical population.

We study a narrow setting: single-turn coding tasks with four author-written personas. This setting provides an objective anchor for comparing correctness and subjective preference, but does not establish cross-domain generalization. It also excludes central vibe-testing settings in which correctness is partial or difficult to verify.

The pipeline also relies on LLMs for rewriting and judging, so rewrites may drift from the intended user framing or task, and LLM judgments may be biased and not fully match human preferences. Prompt personalization changes several factors at once: length, context, constraints, and requested style. Human and LLM judgment agreement on personalized prompts is low and inconclusive, so the observed shifts may partly reflect systematic judge preferences.

Our modular pipeline is simplified by design. Future work can extend it to multi-turn (Liao et al., 2024; Lu et al., 2025) and tool-augmented settings (Chi et al., 2024), strengthen human validation (Kim et al., 2024), replace hand-crafted personas with learned profiles (Davidson et al., 2023; Jiang et al., 2025a; Zhao et al., 2025; Fu et al., 2025; Wang et al., 2025), and expand beyond coding (Li et al., 2024; Chang et al., 2025).

Ethics statement

Our work studies and operationalizes a form of informal evaluation that users already practice. We do not train new language models or deploy a user-facing system; we propose an evaluation pipeline and report controlled experiments.

Human subjects. We collected a survey from volunteer respondents recruited via public social media posts and ran a human annotation task with volunteer annotators. Participation in both was optional and uncompensated. We did not collect sensitive personal data beyond coarse self-reported background and usage habits, and all reported results are aggregated or anonymized. We submitted the study for institutional ethics review as required by our relevant institutions.

In-the-wild sources. Our in-the-wild corpus is derived from publicly available model comparison reports. We use these sources to study evaluation practices rather than to profile individuals. We construct two coarse coding profiles from public Reddit and YouTube material. We extract only task and response preferences relevant to coding assistance, do not infer sensitive attributes, and omit unnecessary personal information from the released profiles. In released artifacts, we retain public source metadata, including names and links to the original materials, and attribute all content to its original authors. All rights to the original materials remain with their respective authors or publishers.

LLM usage and potential harms. Our pipeline uses LLMs for prompt rewriting and for automated judging. These components may encode biases, including preferences for certain writing styles or verbosity, and could disadvantage particular user groups or interaction styles if used without validation. We mitigate position bias via swapped-order judging and include controls for generic paraphrasing, but automated judging remains imperfect. We therefore view the pipeline as a tool for surfacing trade-offs, not a definitive arbiter of model quality.

Acknowledgments

This research was supported by the Israel Science Foundation (grant No. 2942/25), by the Israeli Ministry of Science (grant no. 7256), and the European Union (ERC, Control-LM, 101165402). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency; neither the European Union nor the granting authority can be held responsible for them. We thank the "Google Academic Program Award" for providing access to Gemini. We would like to express our gratitude to Gili Lior for valuable feedback and thoughtful comments throughout this work, and also thank Dana Arad, Tomer Ashuach, Orian Dabod, Noam Dahan, Shahar Levy, Nir Mazor, and Michael Toker for their assistance and support.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Yixin Cao, Shibo Hong, Xinze Li, Jiahao Ying, Yubo Ma, Haiyuan Liang, Yantao Liu, Zijun Yao, Xiaozhi Wang, Dan Huang, et al. Toward generalizable evaluation in the llm era: A survey beyond benchmarks. *arXiv preprint arXiv:2504.18838*, 2025.

Serina Chang, Ashton Anderson, and Jake M Hofman. Chatbench: From static benchmarks to human-ai evaluation. *arXiv preprint arXiv:2504.07114*, 2025.

Yizhou Chi, Lingjun Mao, and Zineng Tang. Amongagents: Evaluating large language models in the interactive text-based social deduction game. *arXiv preprint arXiv:2407.16521*, 2024.

Sam Davidson, Salvatore Romeo, Raphael Shu, James Gung, Arshit Gupta, Saab Mansour, and Yi Zhang. User simulation with large language models for evaluating task-oriented dialogue. *arXiv preprint arXiv:2309.13233*, 2023.

Dave Davies. Evaluating Large Language Models (LLMs): A comprehensive guide for practitioners — online-inference.medium.com. https://tinyurl.com/4ws2jwuy, 2025.

Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzalez. Vibecheck: Discover and quantify qualitative differences in large language models. *arXiv preprint arXiv:2410.12851*, 2024.

Xiao Fu, Hossein A Rahmani, Bin Wu, Jerome Ramos, Emine Yilmaz, and Aldo Lipani. Pref: Reference-free evaluation of personalised text generation in llms. *arXiv preprint arXiv:2508.10028*, 2025.

Google DeepMind. Gemini 3 flash model card, December 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf. Published: December 2025.

Eliya Habba, Noam Dahan, Gili Lior, and Gabriel Stanovsky. PromptSuite: A task-agnostic framework for multi-prompt generation. In Ivan Habernal, Peter Schulam, and Jörg Tiedemann (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 254–263, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-334-0. doi: 10.18653/v1/2025.emnlp-demos.19. URL https://aclanthology.org/2025.emnlp-demos.19/.

huggingFace. Introducing AI Sheets: a tool to work with datasets using open AI models! — huggingface.co. https://huggingface.co/blog/aisheets, 2025.

Naman Jain, Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Representation Learning*, volume 2025, pp. 58791–58831, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/94074dd5a072d28ff75a76dabed43767-Paper-Conference.pdf.

Qi Jia, Xiang Yue, Tianyu Zheng, Jie Huang, and Bill Yuchen Lin. Simulbench: Evaluating language models with creative simulation tasks. In *North American Chapter of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusId:272600407.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*, 2025a.

Hongchao Jiang, Yiming Chen, Yushi Cao, Hung-yi Lee, and Robby T Tan. Codejudgebench: Benchmarking llm-as-a-judge for coding tasks. *arXiv preprint arXiv:2507.10535*, 2025b.

Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram'e, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gael Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Istvan Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, Andr'as Gyorgy, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Boxi Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, Cj Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Pluci'nska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, J. Michael Wieting, Jonathan Lai, Jordi Orbay, Joe Fernandez, Joshua Newlan, Junsong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Ping mei Xu, Piotr Stańczyk, Pouya Dehghani Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Ardeshir Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal

Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vladimir Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab S. Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and L'eonard Hussenot. Gemma 3 technical report. *ArXiv*, abs/2503.19786, 2025. URL https://api.semanticscholar.org/CorpusID:277313563.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.324. URL https://aclanthology.org/2021.naacl-main.324/.

Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. Evallm: Interactive evaluation of large language model prompts on user-defined criteria. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1–21, 2024.

Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, Enamul Hoque, Shafiq Joty, and Jimmy Xiangji Huang. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13785–13816, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.764. URL https://aclanthology.org/2024.emnlp-main.764/.

Ruosen Li, Ruochen Li, Barry Wang, and Xinya Du. Iqa-eval: Automatic evaluation of human-model interactive question answering. *Advances in Neural Information Processing Systems*, 37:109894–109921, 2024.

Yusheng Liao, Yutong Meng, Yuhao Wang, Hongcheng Liu, Yanfeng Wang, and Yu Wang. Automatic interactive evaluation for large language models with state aware patient simulator. *arXiv preprint arXiv:2403.08495*, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Yuxuan Lu, Jing Huang, Yan Han, Bingsheng Yao, Sisong Bei, Jiri Gesi, Yaochen Xie, Qi He, Dakuo Wang, et al. Can llm agents simulate multi-turn human behavior? evidence from real online customer behavior data. *arXiv preprint arXiv:2503.20749*, 2025.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. Duetsim: Building user simulator with dual large language models for task-oriented dialogues. *ArXiv*, abs/2405.13028, 2024. URL https://api.semanticscholar.org/CorpusId:269804692.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, et al. Dataperf:

Benchmarks for data-centric ai development. *Advances in Neural Information Processing Systems*, 36:5320–5347, 2023.

Meta. Llama 3.3 model card, December 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md. Model release date (70B Instruct): 2024-12-06.

OpenAI. Gpt-4o system card, August 2024. URL https://cdn.openai.com/gpt-4o-system-card.pdf.

OpenAI. Sycophancy in GPT-4o: What happened and what we're doing about it — openai.com. https://openai.com/index/sycophancy-in-gpt-4o/, 2025a.

OpenAI. Gpt-5.1 instant and gpt-5.1 thinking system card addendum, November 2025b. URL https://openai.com/index/gpt-5-system-card-addendum-gpt-5-1/.

Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. A new era of intelligence with gemini 3, 2025.

Jon Saad-Falcon, Rajan Vivek, William Berrios, Nandita Shankar Naik, Matija Franklin, Bertie Vidgen, Amanpreet Singh, Douwe Kiela, and Shikib Mehri. Lmunit: Fine-grained evaluation with natural language unit tests. *arXiv preprint arXiv:2412.13091*, 2024.

Manya Wadhwa, Zayne Sprague, Chaitanya Malaviya, Philippe Laban, Junyi Jessy Li, and Greg Durrett. Evalagent: Discovering implicit evaluation criteria from the web. *arXiv preprint arXiv:2504.15219*, 2025.

Kuang Wang, Xianfei Li, Shenghao Yang, Li Zhou, Feng Jiang, and Haizhou Li. Know you first and be you better: Modeling human-like user simulators via implicit profiles. *ArXiv*, abs/2502.18968, 2025. URL https://api.semanticscholar.org/CorpusID:276617942.

Laura Weidinger, Inioluwa Deborah Raji, Hanna Wallach, Margaret Mitchell, Angelina Wang, Olawale Salaudeen, Rishi Bommasani, Deep Ganguli, Sanmi Koyejo, and William Isaac. Toward an evaluation science for generative ai systems. *arXiv preprint arXiv:2503.05336*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Y Zhang, Y Mai, JSR Roberts, R Bommasani, Y Dubois, and P Liang. Helm instruct: a multidimensional instruction following evaluation framework with absolute ratings, 2024.

Zheng Zhao, Clara Vania, Subhradeep Kayal, Naila Khan, Shay B Cohen, and Emine Yilmaz. Personalens: A benchmark for personalization evaluation in conversational ai assistants. *arXiv preprint arXiv:2506.09902*, 2025.

Ming Zhong, Xiang Zhou, Ting-Yun Chang, Qingze Wang, Nan Xu, Xiance Si, Dan Garrette, Shyam Upadhyay, Jeremiah Liu, Jiawei Han, et al. Vibe checker: Aligning code evaluation with human preference. *arXiv preprint arXiv:2510.07315*, 2025.

A Survey extended details

We recruited volunteer respondents via public social media posts aimed at a technical audience (e.g., X and Reddit). Participation was optional and uncompensated. We collected coarse self-reports of technical background and usage habits, as well as questions about evaluation routines, perceived benchmark gaps, and interest in automation. Before answering any questions, participants were shown the information and definition as in Figure 10. In the released survey results, examples of open-ended prompts and "golden prompts" were omitted to preserve respondent privacy. The full survey questions and answer distributions are in Table 21, Table 22, Table 23, and Table 24.

Some questions were optional or multi-select. We therefore compute percentages with respect to the number of respondents who answered each question (or selected at least one option in multi-select questions). Likert-style questions are reported as distributions over the 1–7 scale. When reporting means in the main text, we compute them from the raw counts; due to rounding in percentage displays, means computed from the rounded percentages may differ slightly.

Figure 2 in the main paper visualizes two multiple-choice questions from Table 22 and Table 24: (i) perceived benchmark failures (Q11), and (ii) common vibe-testing methods (Q7). We plot the same percentages, ordered by frequency, and use the table above as the authoritative reference for exact values.

B In-the-wild examples labeling

We provide a detailed description of the four-stage procedure summarized in Section 3, including the prompting setup used for extraction and labeling, the dimension consolidation process, and the final corpus-level analysis.

B.1 Goal and scope

The goal of this analysis is to empirically ground our definition of *vibe-testing* by documenting how practitioners qualitatively evaluate LLMs in real-world settings. Accordingly, the corpus is curated for concreteness and diversity rather than representativeness, and the analysis is qualitative and exploratory.

B.2 Stage (1): source retrieval and selection

We manually collected 40 public model comparison reports from four source types: YouTube review transcripts, Reddit threads, blog posts, and technology news articles. Sources were identified via targeted searches combining model names (e.g., "GPT", "Claude", "Gemini"), comparison terms (e.g., "vs", "review", "comparison"), and subjective language (e.g., "feels", "vibes", "I prefer", "works better for me"). See source list in Table 19 and Table 20.

We included sources that satisfy the three criteria stated in the main text: (i) reference at least one specific model, (ii) describe at least one concrete test input (prompt, task, or scenario), and (iii) include qualitative judgments and subjective claims about model behavior or output quality. We excluded sources that were purely promotional, focused exclusively on benchmark scores without qualitative discussion, or lacked concrete test inputs. The final set of 40 sources was selected to maximize diversity of task types and interaction settings.

B.3 Stage (2): vibe-test extraction and structured labeling

For each source, we used LLMs (GPT-5.2 and Gemini 3 Pro) to extract and label *vibe-tests*, defined as localized *instances* where an author evaluates one or more models on a specific task using subjective criteria. For both YouTube transcripts and Reddit threads, the LLM was prompted to (a) produce a brief analysis identifying candidate vibe-testing instances and checking them against the definition, and then (b) output a JSON array of extracted vibe-tests with structured fields.

Each extracted vibe-test included:

- `quote`: a direct excerpt capturing the instance (1–5 sentences);
- `task_type`: a brief description of the tested task or scenario;
- `models_mentioned`: a list of model names mentioned (empty if not specified);
- `vibe_language`: the subjective descriptors used in the quote;
- `why_this_is_vibe_testing`: a short justification;
- `benchmark_mention`: whether benchmarks were mentioned (`Yes/No`);
- `timestamp_range` (YouTube) or `metadata` when available (Reddit).

The authors manually verified the sampled extracted vibe-tests and iterated over the prompt to remove false positives (e.g., general opinions without a concrete task, or purely benchmark-driven comparisons), corrected extraction errors (e.g., truncated quotes, misattributed models), and retained only instances that clearly satisfy the operational definition.

LLM prompt for vibe-test extraction and labeling. We used the prompt in Figure 11 (with minor formatting adjustments between YouTube and Reddit to reflect available metadata):

B.4 Stage (3): trend analysis and common dimensions

Stage (3) produces a consolidated, closed set of recurring subjective dimensions and applies it consistently across all vibe-tests.

(3a) Proposing and reconciling common dimensions. Starting from the open-ended `vibe_language` and justifications produced in Stage (2), we asked the LLMs to propose lists of repeated subjective dimensions appearing across vibe-tests, grouping similar criteria under shared labels. In parallel, we independently compiled candidate lists from manual review. We then iteratively reconciled and refined these lists, merging overlapping categories, splitting overly broad ones, and discarding idiosyncratic or weakly supported dimensions, to derive a final closed set of common dimensions.

(3b) Dimension-based re-annotation of each vibe-test. After fixing the closed set of dimensions, we performed a second annotation pass in which each vibe-test was re-annotated using only this dimension list. We prompted the model to output a single JSON object per vibe-test, selecting dimensions only when clearly supported by the text and providing per-dimension justifications citing exact phrases. The model was explicitly instructed to avoid inferring missing information and to mark missing evidence as `not stated` or `uncertain`. We compiled the resulting per-instance annotations into a single consolidated JSON file representing the full in-the-wild corpus.

Prompt for dimension-based re-annotation. We used the prompt in Figure 12 to re-annotate each vibe-test with the fixed dimension set:

B.5 Stage (4): framework consistency check and gap analysis

Finally, we provided (i) the draft paper definitions (including the proposed input/output dimensions) and (ii) the consolidated JSON of dimension-annotated vibe-tests to an LLM, and asked it to conduct a critical consistency check between the framework and the empirical data.

The model was instructed to:

- verify that the paper definitions encompass the subjective language used in the vibe-tests,
- verify that the dimension inventory covers all distinct "vibes" found in the corpus,
- identify **unmapped instances** whose language does not fit the proposed dimensions,

Input Dimension	Operational Meaning	Typical Cues
Task type	The kind of task used as the test, ranging from short, isolated prompts to multi-step scenarios (e.g., single-turn coding, debugging, planning, writing, workflow tasks).	Mentions of "my daily workflow," "I tried debugging," "I asked it to plan," or explicit task categories (code generation, refactor, summarization, reasoning).
Task complexity	How demanding the test is in length, number of constraints, required steps, and need for long-range coherence.	Short one-shot prompts vs. long prompts with many requirements, multi-step instructions, or tasks requiring sustained structure.
Real-world context	Whether the prompt situates the task in a concrete domain or scenario (personal, professional, or everyday), rather than an abstract benchmark-like query.	"In my codebase," "for my project," "as a product manager," "explain like cooking," domain-specific details, constraints, or artifacts.
Persona-based framing	How the user frames the interaction style, role, or audience (tutor vs. peer, novice vs. expert, tone requirements).	"Explain to a beginner," "be concise," "act as a senior engineer," "write like a legal memo," explicit tone or audience constraints.
Underspecification level	How much the prompt leaves implicit, to test the model's ability to ask clarifying questions, make reasonable assumptions, or handle ambiguity.	Vague instructions on purpose ("do something better"), missing details, open-ended tasks, or explicit stress-tests of ambiguity handling.
Constraint tightness	How strictly the prompt constrains format, tools, style, or rules (schemas, word limits, citation rules, refusal constraints).	"Return JSON with this schema," "no markdown," "must cite sources," "do not use X," strict formatting or policy constraints.
Reference availability	Whether the prompt provides grounding material (code, document excerpt, table) vs. asking from scratch.	Presence of pasted context, files, or citations, vs. prompts that rely on the model's background knowledge alone.

Table 2: **Taxonomy of vibe-testing dimensions.** We organize recurring axes of variation into input-oriented dimensions (what users choose to test and how they frame it) and output-oriented dimensions (how users compare outputs and what qualities they prioritize). Each dimension is intended to be actionable: it can be instantiated when constructing tests and referenced when judging model responses.

- identify **unsupported theory** (dimensions defined in the paper but absent from the corpus),
- highlight **mismatches** where our definitions conflict with practitioner usage.

We used this output as a gap analysis to refine dimension definitions and ensure that the final framework closely reflects the empirical examples rather than purely theoretical assumptions.

Prompt for consistency check and gap analysis. For the final analysis, we used the prompt in Figure 13 (with the paper definitions and the consolidated JSON provided as inputs):

> You will be provided with (1) the draft paper definitions and (2) a JSON of labeled vibe-testing instances. Your objective is to conduct a rigorous consistency check and gap analysis between the theoretical framework and the empirical data: verify that the definitions encompass the subjective language used, verify coverage of all distinct vibes, identify instances that do not fit the proposed dimensions, and identify paper dimensions that lack supporting evidence. Be critical and use direct quotes from the instances to support every critique.

B.6 Vibe dimensions details

The detailed list of input and output vibe dimensions is shown in Tables 2 and Table 3.

Output Dimension	Operational Meaning	Typical Cues
Comparison setup	How the user elicits and compares outputs, including side-by-side comparisons, reruns, and multi-prompt probing.	"I compared A vs. B," "ran it twice," "tried a few variants," "it was inconsistent across runs."
Correctness	Whether the output is correct with respect to the task requirements and stated constraints.	Explicit correctness judgments ("it works," "it is wrong," "hallucinated"), tests with executable code, or fact-check claims.
Clarity	How readable and well organized the response is for the intended user, including logical flow and formatting.	Mentions of "easy to follow," "well structured," "confusing," "jumps around," "good explanation."
Cognitive load	How mentally taxing the response is, given chunking, ordering, and amount of new concepts introduced at once.	"Too much at once," "overwhelming," "nice step-by-step," "dumped details," "hard to track."
Style/Tone fit	Alignment with the requested or expected tone, persona, and communication style for the user context.	"Too formal," "felt friendly," "matches my style," "sounds like marketing," "too robotic."
Workflow fit	Whether the output integrates into the user's actual workflow, helping them complete the task with minimal friction.	"I can paste this into my codebase," "saved me time," "usable," "does not match how I work," "not actionable."
Friction	Whether the model resists the user's intent in frustrating ways (fighting constraints, derailing from role, refusing incorrectly, or being hard to steer).	"It kept insisting," "ignored constraints," "would not do what I asked," "felt gaslighting," "hard to steer."
Ambiguity handling	How the model behaves under underspecified inputs, including making reasonable assumptions and asking clarifying questions.	"Asked the right questions," "made bad assumptions," "filled in details incorrectly," "handled vagueness well."
Reliability	Consistency across reruns and robustness to small prompt changes, including sensitivity and variance.	"Sometimes it works," "unstable," "high variance," "breaks with tiny changes," "consistent across attempts."
Trustworthiness	Whether the model behaves safely and predictably, including refusal correctness and avoidance of harmful or risky guidance.	Mentions of unsafe suggestions, unnecessary refusals, policy overreach, or trustworthy caution (especially on sensitive tasks).
Anthropomorphism	Whether the interaction feels human-like vs. robotic, including naturalness, conversational coherence, and perceived "personality."	"Felt human," "felt robotic," "sounds like a real assistant," "too template-y," "has a vibe."

Table 3: **Taxonomy of vibe-testing output dimensions: how users compare outputs and what they value in responses.** We organize recurring axes of variation into input-oriented dimensions (what users choose to test and how they frame it) and output-oriented dimensions (how users compare outputs and what qualities they prioritize). Each dimension is intended to be actionable: it can be instantiated when constructing tests and referenced when judging model responses.

C Experimental Details

We provide full reproducibility details, including model inference settings, persona specifications, prompt templates, the evaluated dimension subset with guidance, and the judging and debiasing protocol.

Model inference settings. For each model, we report: (i) system prompt, (ii) decoding parameters (e.g., temperature/top-p/top-k or greedy), (iii) token limits, and any provider-specific options. For GPT-5.1, we use model ID `gpt-5.1-2025-11-13` and for Gemini-3-Pro `gemini-3-pro-preview`. All GPT models run with thinking enabled when relevant (low effort) and with a $5,000$ token limit, and Qwen3 models use a $15,000$-token limit to accommodate longer reasoning traces.

Persona profiles. We provide the full persona specifications (YAML/JSON): persona description (Also in Table 4), input-dimension settings, output-dimension weights (1–5), and expressive-style instructions used by the prompt composer and judge.

Prompt variants. We include the exact templates for: (i) the original dataset prompt formatting, (ii) the personalized prompt rewriting prompts used by GPT-5.1 and Qwen3-32B, and (iii) the control prompt generation procedure and templates for PROMPTSUITE. We also document how variants are constructed for each persona.

Vibe dimensions used for coding. We list the final subset of vibe dimensions evaluated in coding assistance, with brief guidance text for each dimension as used by the judge.

Judging protocol and de-biasing. We provide the full judge prompt, the required output format (A/B/TIE plus rationale and confidence), and the position-swap procedure. We specify the confidence-based conflict resolution rule and the alternative agreement-only rule used as a robustness check.

Regarding LLM judge model choice – pilots with Llama3.3-70B (Meta, 2024) and Gemini 3 Flash (Google DeepMind, 2025) yielded similar or lower-quality judgments and are omitted.

C.1 Prompts and personas used in the pipeline

We release the full prompts used across all stages of the pipeline: persona parsing, prompt-change proposal, personalized prompt generation and verification, and subjective pairwise judging.

C.1.1 Personas

The pipeline evaluates models under four fixed user personas that span a novice-to-expert spectrum (Section 6.2). Each persona is represented as a structured profile with (i) a short natural-language description and (ii) explicit preferences over the input and output dimensions. Profiles are used in two places: (1) to personalize benchmark prompts while preserving the underlying task, and (2) to condition pairwise subjective judging on what a given persona values.

In our release, persona profiles are stored as JSON/YAML files, one per persona, with a shared schema. Each profile includes a `persona_description` field, an `input_dimensions` object, and an `output_dimensions` object whose entries are either categorical labels (for discrete dimensions) or importance weights on a 1–5 scale (for prioritized output criteria). See the personas description in Table 4.

Persona Creator We include in our code a *Persona Creator* - a tool that converts unstructured user information into the structured profile format used by the pipeline. It accepts direct user descriptions, survey responses, public posts, and other collections of user-provided text. The output follows the profile schema used in our experiments, including input dimensions for prompt personalization, output dimensions for evaluation, and optional importance weights over output dimensions. The Persona Creator extracts only preferences supported by the source material. Unsupported fields are left unspecified to avoid making assumptions about the user. Generated profiles can be manually reviewed and edited before use. See our code for the exact generation prompt, output schema, and example configurations.

C.1.2 Prompts used in the pipeline

We release the full prompts used across all stages of the pipeline: persona parsing, prompt-change proposal, personalized prompt generation, verification of semantic preservation, and model evaluation and judging. Prompts are stored in configuration files, with model-specific wrappers (e.g., provider system and developer messages) and stage-specific user prompts. Below, we provide representative templates for the main stages. Throughout, we enforce strict output formatting to support reliable automation and downstream parsing.

Profile	Description and Preference Weights
Beginner Student	"I am an undergraduate student with limited Python experience, mainly using it for basic data analysis in my statistics courses. I need code that is very clear and well-explained." *Weights:* Clarity: 5, Style: 1, Workflow Fit: 2, Cognitive Load: 1, Context Awareness: 4, Persona Consistency: 4, Anthropomorphism: 5.
Intermediate Learner	"I have moderate coding experience and typically work on debugging and small projects. I care most about getting the correct solution, and I appreciate helpful clarifications that support my learning as I go." *Weights:* Clarity: 5, Style: 1, Workflow Fit: 3, Cognitive Load: 1, Context Awareness: 4, Persona Consistency: 4, Anthropomorphism: 3.
AI Researcher	"I am a researcher in the field of machine learning. I need code that is efficient and easy to modify and trace." *Weights:* Clarity: 4, Style: 1, Workflow Fit: 4, Cognitive Load: 1, Context Awareness: 4, Persona Consistency: 4, Anthropomorphism: 1.
Advanced Developer	"I am an experienced programmer and I often work on optimization and complex debugging. I prefer efficient solutions with minimal guidance and concise outputs." *Weights:* Clarity: 3, Style: 1, Workflow Fit: 5, Cognitive Load: 1, Context Awareness: 4, Persona Consistency: 4, Anthropomorphism: 1.

Table 4: Representative user persona used to instantiate personalized vibe-testing in coding assistance. Each persona includes a short natural-language description and importance weights for the output dimensions used in evaluation. These descriptions and weights were chosen before running experiments and were not tuned in any way.

Provider wrapper. For models that support developer messages, we prepend a minimal wrapper to encourage strict adherence to instructions.

> **Developer message:** Follow the instructions strictly.

For the Qwen3 models, we used the recommended default instruction with thinking enabled:

> **Developer message:** You are a helpful assistant. Please first think about the question thoroughly. Consider multiple approaches and show your reasoning. Wrap your thinking in <think> and </think> tags and then return your final answer.

Persona parsing. Given a short natural-language user description, the persona-parsing prompt asks an LLM to produce a structured JSON profile describing input and output preferences. The model must output a single JSON object and nothing else (Prompt is in Figure 14).

Change identification (profile → concrete prompt modifications). To operationalize a persona profile into actionable prompt edits, we ask an LLM to propose 2–3 concrete modification options for a fixed set of fields, while explicitly disallowing changes that alter the task itself. This stage outputs a single JSON object with a list of changes keyed by profile fields (prompt is in Figure 15).

Personalized prompt composition. Given an original benchmark prompt and the selected modifications, we generate a personalized version that preserves the underlying programming task. The prompt is written in the persona voice (first person), avoids explicit references to the profile schema, and is constrained to a short length (prompt is in Figure 16).

HumanEval+ prefix composition. For HumanEval+ style prompts that include code context and docstrings, we produce only a short persona prefix that is concatenated to the original prompt (Figure 17). This avoids perturbing code formatting while still injecting persona-relevant framing.

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.03* (0.01)	0.02* (0.01)	0.12* (0.02)	**0.85*** (0.00)
vs.	Personalized	**0.60*** (0.05)	**0.63*** (0.03)	0.39* (0.01)	**0.63*** (0.03)
GPT-OSS-20B	Control	0.12* (0.01)	0.10* (0.01)	0.20* (0.02)	**0.87*** (0.01)
GPT-5.1	Original	0.13* (0.01)	0.22* (0.03)	**0.68*** (0.03)	**0.92*** (0.01)
vs.	Personalized	**0.90*** (0.01)	**0.90*** (0.02)	**0.85*** (0.01)	**0.85*** (0.02)
GPT-4o	Control	0.11* (0.01)	0.12* (0.04)	**0.66*** (0.03)	**0.92*** (0.01)

Table 5: **Personalization shifts win-rates (HumanEval+).** Per-sample win rates for each model pair are shown by persona and prompt type; win-rates are reported from the perspective of the first model in each pair. Preference shifts mirror the ones observed in MBPP+: GPT-5.1 underperforms on the original Beginner/Intermediate prompt, but wins on personalized prompts. GPT-5.1 remains stronger for the Advanced persona in all prompts. * denotes statistical significance (two-sided binomial test).

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.04* (0.00)	0.01* (0.01)	0.03* (0.02)	**0.82*** (0.00)
vs.	Personalized	0.40* (0.03)	**0.55*** (0.05)	0.48 (0.12)	0.45* (0.10)
GPT-OSS-20B	Control	0.04* (0.00)	0.02* (0.01)	0.03* (0.02)	**0.84*** (0.01)
Qwen3-32B	Original	0.56 (0.10)	**0.58*** (0.08)	**0.61*** (0.09)	0.46 (0.07)
vs.	Personalized	**0.64*** (0.01)	**0.63*** (0.07)	**0.61*** (0.08)	0.41 (0.17)
Qwen3-14B	Control	0.55 (0.04)	0.54 (0.07)	0.45 (0.12)	0.44* (0.08)

Table 6: **Personalization shifts remain similar with Qwen3-32B as generator.** Per-sample win rates on MBPP+ by user and prompt type, reported for the first model in each pair (tie rate in parentheses), when personalized rewrites are generated by Qwen3-32B instead of GPT-5.1. Overall trends largely align with the main results, suggesting that the observed preference shifts are not specific to any single generator. One difference is a smaller increase in win-rate for GPT-5.1 on the Beginner and Advanced personas, possibly due to different judge sets. Judges are Qwen3-14B and GPT-OSS-20B for GPT-5.1 *vs.* GPT-OSS-20B, and GPT-OSS-20B for Qwen3-32B *vs.* Qwen3-14B. * denotes statistical significance on a two-sided binomial test.

Semantic-preservation verification. To ensure personalized prompts remain faithful to the original benchmark intent, we use a verifier prompt that checks (i) whether the end goal is identical and (ii) whether the ground-truth solution set is preserved. The verifier returns two booleans and an error string if either check fails (prompt is in Figure 18).

Notes on implementation and release. All prompt templates above are parameterized using placeholders (e.g., `{original_prompt}`) and are instantiated deterministically by the pipeline. We release the full, exact prompt texts (including system and developer wrappers), together with the JSON schemas used for validation, in the accompanying repository to support faithful reproduction.

D Additional results

We report additional results to test the robustness of the main findings along three axes: a second benchmark (HumanEval+), a different personalized prompt generator, and alternative aggregation rules. Figures 7, 8, and 9 show the same per-dimension breakdown across

Grouping	Subset	Agreement (%)	Fleiss's κ
Model pair	GPT-5.1 *vs.* GPT-OSS-20B	0.82 ± 0.14	0.38 ± 0.21
	GPT-5.1 *vs.* GPT-4o	0.85 ± 0.10	0.38 ± 0.12
	Gemini-3-Pro *vs.* Gemma-3-4B	0.82 ± 0.07	0.49 ± 0.13
	Qwen3-32B *vs.* Qwen3-14B	0.64 ± 0.07	0.30 ± 0.12
Persona	Beginner	0.81 ± 0.13	0.36 ± 0.10
	Intermediate	0.78 ± 0.13	0.45 ± 0.18
	Researcher	0.74 ± 0.14	0.28 ± 0.17
	Advanced	0.79 ± 0.11	0.45 ± 0.13
Prompt type	Original	0.79 ± 0.14	0.43 ± 0.22
	Control	0.78 ± 0.13	0.41 ± 0.18
	Personalized	0.77 ± 0.12	0.34 ± 0.09
Judge pair	GPT-5.1 *vs.* GPT-OSS-20B	0.78 ± 0.14	0.38 ± 0.18
	GPT-5.1 *vs.* Qwen3-14B	0.79 ± 0.14	0.40 ± 0.20
	Qwen3-14B *vs.* GPT-OSS-20B	0.77 ± 0.13	0.38 ± 0.19
Overall	All Samples	$\mathbf{0.78} \pm 0.13$	$\mathbf{0.39} \pm 0.16$

Table 7: LLM-judge agreement for the subjective pairwise preference labels, reported over multiple pooled slices of the evaluation. For each subset, we aggregate all its pairwise decisions and compute (i) **raw agreement** as the mean percentage of items on which two judges output the same label, and (ii) **Fleiss's** κ, which adjusts agreement for chance given the judges' marginal label distributions (note that when one model is selected as the winner in most items, the resulting label imbalance can lower κ despite high raw agreement). Values are reported as mean $\pm$ std, where the mean and standard deviation are computed across the judges' pairs when available.

personas as Figure 4, for the remaining three model pairs on the main results. Across all settings, the main qualitative pattern remains the same: personalized rewrites often change model preferences, while neutral control paraphrases largely preserve the original pattern.

We report additional results to test the robustness of the main findings along three axes: a second benchmark (HumanEval+), a different personalized prompt generator, and alternative aggregation rules. Across all settings, the main qualitative pattern remains the same: personalized rewrites often change model preferences, while neutral control paraphrases largely preserve the original pattern.

HumanEval+ shows the same qualitative trend. Table 5 reports head-to-head results on HumanEval+ for two model pairs, GPT-5.1 *vs.* GPT-4o and GPT-5.1 *vs.* GPT-OSS-20B, under the same evaluation protocol used in the main experiments. The results broadly mirror MBPP+. For GPT-5.1 *vs.* GPT-4o, GPT-5.1 underperforms on original prompts for the Beginner and Intermediate personas, but becomes strongly preferred under personalized prompts across all personas. Control paraphrases remain close to the original pattern. For GPT-5.1 *vs.* GPT-OSS-20B, GPT-5.1 is strongly disfavored on original prompts for the Beginner, Intermediate, and Researcher personas, but personalized rewrites substantially improve its win rate, making the comparison much more balanced for Beginner, Intermediate, and Advanced. Due to inference cost, HumanEval+ was restricted to these two model pairs, so we treat it as a supporting robustness check rather than a comprehensive replication across all model families.

Using Qwen3-32B as the prompt generator yields mostly similar shifts. Table 6 repeats the MBPP+ analysis with personalized rewrites generated by Qwen3-32B instead of GPT-5.1. The overall trends remain similar to the main results. For GPT-5.1 *vs.* GPT-OSS-20B, personalization still substantially improves GPT-5.1's standing relative to original and control

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.09* (0.02)	0.15* (0.07)	**0.61*** (0.06)	**0.87*** (0.04)
vs.	Personalized	**0.92*** (0.03)	**0.75*** (0.07)	**0.93*** (0.04)	**0.74*** (0.06)
GPT-4o	Control	0.08* (0.03)	0.18* (0.07)	**0.69*** (0.06)	**0.93*** (0.03)
GPT-5.1	Original	0.04* (0.02)	0.01* (0.02)	0.08* (0.05)	**0.91*** (0.02)
vs.	Personalized	**0.73*** (0.06)	0.53 (0.10)	0.38* (0.09)	0.53 (0.07)
GPT-OSS-20B	Control	0.06* (0.04)	0.03* (0.02)	0.09* (0.08)	**0.93*** (0.03)
Gemini-3-Pro	Original	0.45 (0.07)	0.50 (0.08)	0.46 (0.07)	**0.65*** (0.03)
vs.	Personalized	**0.92*** (0.02)	**0.85*** (0.05)	**0.95*** (0.02)	**0.56*** (0.09)
Gemma-3-4B	Control	0.39* (0.07)	0.44* (0.06)	0.45* (0.08)	**0.67*** (0.05)
Qwen3-32B	Original	**0.44*** (0.13)	0.47 (0.17)	**0.44*** (0.16)	0.35* (0.09)
vs.	Personalized	0.49 (0.09)	**0.60*** (0.11)	0.52 (0.13)	**0.46*** (0.10)
Qwen3-14B	Control	**0.45*** (0.15)	0.50 (0.14)	0.46 (0.15)	0.37* (0.11)

Table 8: **Similar trends with unweighted aggregation.** Per-sample win rates on the same MBPP+ comparisons as in the main results, but with equal weight assigned to all output dimensions. Results remain close to the main findings, suggesting that the observed preference shifts are already strong without persona-specific weighting. * denotes statistical significance on a two-sided binomial test.

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.09* (0.00)	0.16* (0.02)	**0.63*** (0.02)	**0.88*** (0.00)
vs.	Personalized	**0.94*** (0.01)	**0.77*** (0.02)	**0.97*** (0.00)	**0.82*** (0.02)
GPT-4o	Control	0.08* (0.01)	0.19* (0.03)	**0.70*** (0.02)	**0.95*** (0.01)
GPT-5.1	Original	0.03* (0.00)	0.01* (0.00)	0.08* (0.02)	**0.91*** (0.01)
vs.	Personalized	**0.77*** (0.02)	**0.55*** (0.02)	0.43* (0.01)	**0.58*** (0.03)
GPT-OSS-20B	Control	0.06* (0.00)	0.02* (0.00)	0.10* (0.02)	**0.93*** (0.01)
Gemini-3-Pro	Original	0.48 (0.02)	0.53 (0.01)	0.48 (0.03)	**0.66*** (0.02)
vs.	Personalized	**0.93*** (0.01)	**0.90*** (0.01)	**0.97*** (0.01)	**0.70*** (0.02)
Gemma-3-4B	Control	0.44* (0.02)	0.45* (0.02)	0.48 (0.02)	**0.69*** (0.02)
Qwen3-32B	Original	0.54 (0.04)	**0.62*** (0.03)	0.53 (0.04)	0.38* (0.05)
vs.	Personalized	0.54 (0.02)	**0.67*** (0.04)	**0.62*** (0.03)	**0.54*** (0.04)
Qwen3-14B	Control	0.54 (0.05)	**0.59*** (0.04)	**0.55*** (0.06)	0.40* (0.05)

Table 9: **Similar trends when correctness is not used to determine the sample-level winner.** Per-sample win rates on the same MBPP+ comparisons as in the main results, but with winners determined only from dimension-level judgments, ignoring correctness. Results remain similar, suggesting that the main preference patterns are not driven primarily by the correctness gate. * denotes statistical significance on a two-sided binomial test.

prompts, especially for Intermediate, and shifts Qwen3-32B *vs.* Qwen3-14B preferences toward the larger model except for the Advanced persona. One difference is that the personalized advantage for GPT-5.1 over GPT-OSS-20B is weaker for the Beginner and Advanced personas, where win rates fall below 0.50. A likely reason is that, unlike the main setup, this study does not include GPT-5.1 as a judge due to cost; in the main results, GPT-5.1 judgments

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.08* (0.01)	0.16* (0.03)	**0.61*** (0.04)	**0.86*** (0.01)
vs.	Personalized	**0.93*** (0.01)	**0.75*** (0.03)	**0.95*** (0.01)	**0.82*** (0.03)
GPT-4o	Control	0.07* (0.01)	0.17* (0.04)	**0.68*** (0.04)	**0.94*** (0.01)
GPT-5.1	Original	0.04* (0.00)	0.01* (0.00)	0.07* (0.06)	**0.92*** (0.01)
vs.	Personalized	**0.76*** (0.03)	**0.57*** (0.04)	0.37* (0.03)	**0.56*** (0.04)
GPT-OSS-20B	Control	0.06* (0.01)	0.02* (0.01)	0.10* (0.07)	**0.94*** (0.01)
Gemini-3-Pro	Original	0.48 (0.03)	0.53 (0.02)	0.49 (0.03)	**0.68*** (0.02)
vs.	Personalized	**0.93*** (0.01)	**0.88*** (0.02)	**0.97*** (0.01)	**0.70*** (0.02)
Gemma-3-4B	Control	0.44* (0.03)	0.45* (0.03)	0.47 (0.04)	**0.69*** (0.04)
Qwen3-32B	Original	0.48 (0.10)	**0.56*** (0.09)	**0.42*** (0.20)	0.35* (0.18)
vs.	Personalized	0.53 (0.05)	**0.68*** (0.08)	**0.61*** (0.04)	0.53 (0.08)
Qwen3-14B	Control	0.50 (0.10)	0.52 (0.12)	**0.42*** (0.17)	0.36* (0.18)

Table 10: **Similar results without confidence-based tie breaking.** Per-sample win rates on the same MBPP+ comparisons as in the main results, but treating swapped-order disagreements as ties rather than resolving them with judge confidence. Trends remain similar, indicating that position effects have only a small impact on the overall results. * denotes statistical significance on a two-sided binomial test.

were more favorable to GPT-5.1 on these personas. This suggests that the size of the shift is somewhat sensitive to the judge set, even though its overall direction remains similar.

Unweighted aggregation yields similar results. Table 8 reports results when all output dimensions are given equal weight, rather than persona-specific importance weights. The main qualitative trends remain intact. Personalized prompts still strongly improve GPT-5.1's win rate against both GPT-4o and GPT-OSS-20B, and similarly strengthen Gemini-3-Pro relative to Gemma-3-4B. For Qwen3-32B *vs.* Qwen3-14B, the shifts remain smaller but generally point in the same direction. This suggests that the preference changes we observe are already strong at the level of the underlying dimension judgments, and do not depend critically on persona-specific weighting, though weighting may still matter in closer comparisons or for users with more extreme priorities.

Ignoring correctness also yields similar trends. Table 9 reports results when sample-level winners are determined directly from dimension judgments, without first using correctness as a gate. Again, the main patterns remain similar. Personalized prompts continue to sharply improve GPT-5.1's standing against GPT-4o and GPT-OSS-20B, and strongly favor Gemini-3-Pro over Gemma-3-4B. For Qwen3-32B *vs.* Qwen3-14B, personalization still shifts preferences toward the larger model for some personas, while leaving others relatively balanced. The similarity to the main results suggests that the observed preference shifts are not driven primarily by the correctness gate, either because most samples are solved correctly or because models that better fit the user also tend to perform better overall.

Treating jointly correct samples as ties. To test whether the observed preference shifts are driven by samples where both models are correct, we repeat the analysis while assigning a tie whenever both models pass the benchmark tests. The remaining samples, where at most one model passes, are evaluated using the same persona-conditioned preference procedure. As expected, this produces high tie rates because both models solve many samples (Table 13). Personalized prompts still reduce ties and shift win rates for the GPT and Qwen comparisons. For Gemini-3-Pro vs. Gemma-3-4B, Gemini's win rate also increases from approximately 0.13–0.14 on original prompts to 0.23–0.41 on personalized prompts, but the comparison remains dominated by ties and the shift is weaker than in the main

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.09* (0.00)	0.16* (0.02)	**0.72*** (0.02)	**0.91*** (0.00)
vs.	Personalized	**0.96*** (0.00)	**0.78*** (0.02)	**0.98*** (0.00)	**0.84*** (0.02)
GPT-4o	Control	0.06* (0.01)	0.15* (0.02)	**0.79*** (0.02)	**0.94*** (0.01)
GPT-5.1	Original	0.05* (0.00)	0.04* (0.00)	0.06* (0.00)	**0.94*** (0.00)
vs.	Personalized	**0.80*** (0.06)	0.55 (0.01)	0.32* (0.15)	**0.58*** (0.04)
GPT-OSS-20B	Control	0.04* (0.00)	0.02* (0.00)	0.05* (0.01)	**0.94*** (0.02)
Gemini-3-Pro	Original	0.48 (0.01)	0.58 (0.00)	0.52 (0.00)	**0.70*** (0.02)
vs.	Personalized	**0.94*** (0.01)	**0.93*** (0.01)	**0.99*** (0.00)	**0.72*** (0.02)
Gemma-3-4B	Control	0.42* (0.02)	0.47 (0.01)	0.48 (0.01)	**0.69*** (0.01)
Qwen3-32B	Original	0.59 (0.02)	**0.64*** (0.03)	0.54 (0.02)	0.40 (0.04)
vs.	Personalized	0.54 (0.01)	**0.70*** (0.03)	**0.65*** (0.01)	0.56 (0.05)
Qwen3-14B	Control	0.53 (0.05)	**0.62*** (0.04)	0.54 (0.06)	0.43 (0.04)

Table 11: **Similar results with majority-vote aggregation across judges.** Per-sample win rates on the same MBPP+ comparisons as in the main results, but using a single majority-vote label per sample instead of counting each judge's vote separately. Trends remain similar, indicating that the main findings are not driven by the original judge aggregation rule. * denotes statistical significance on a two-sided binomial test.

results. The control therefore shows that the personalization effect extends beyond jointly correct samples in several matchups, although it does not appear uniformly across models.

Strict tie handling has little effect. Table 10 reports results when disagreements between swapped response orders are treated as ties, rather than resolved with judge confidence. The resulting win rates remain close to the main results. Personalized prompts still reverse or substantially shift the original preference pattern in the GPT-5.1 and Gemini-3-Pro comparisons, while the Qwen comparison remains weaker and more balanced. This suggests that the main findings are not driven by the confidence-based soft tie-breaker and that residual position effects have only a limited impact on the overall conclusions.

Restricting to disjoint judges preserves the main pattern. To reduce possible self-preference in model judging, we recompute the results using only disjoint judges, excluding votes from judges that are also one of the compared models. The overall trends remain similar to the main results (Table 12). The main exception is the Advanced persona, where the win rate falls to 0.40. This indicates some sensitivity to the judge set, and suggests that majority voting or larger disjoint judge pools would improve robustness.

Majority-vote aggregation also preserves the main pattern. Instead of counting each judge vote separately, we also compute one majority-vote label per sample across judges, with mixed cases resolved conservatively toward ties. The resulting trends are again similar to the main results (Table 11). This suggests that the overall findings are stable to the choice of judge aggregation.

Taken together, these additional analyses strengthen the main conclusion of the paper. The preference shifts induced by personalized prompts generalize beyond MBPP+ to HumanEval+, remain visible when using a different prompt generator, and are robust to several alternative aggregation choices. Across all these checks, the same broad pattern remains: personalization can materially change which model is preferred, whereas neutral paraphrases usually do not.

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.14* (0.00)	0.21* (0.01)	**0.60*** (0.01)	**0.87*** (0.00)
vs.	Personalized	**0.93*** (0.01)	**0.76*** (0.01)	**0.96*** (0.00)	**0.75*** (0.03)
GPT-4o	Control	0.09* (0.01)	0.22* (0.04)	**0.65*** (0.03)	**0.94*** (0.00)
GPT-5.1	Original	0.07* (0.00)	0.04* (0.00)	0.04* (0.00)	**0.88*** (0.02)
vs.	Personalized	**0.72*** (0.04)	0.53 (0.03)	0.20* (0.03)	0.40* (0.04)
GPT-OSS-20B	Control	0.09* (0.00)	0.02* (0.00)	0.02* (0.00)	**0.90*** (0.01)
Qwen3-32B	Original	0.55 (0.02)	**0.62*** (0.03)	0.49 (0.03)	0.40* (0.02)
vs.	Personalized	**0.56*** (0.01)	**0.66*** (0.04)	**0.60*** (0.03)	0.52 (0.04)
Qwen3-14B	Control	**0.56*** (0.05)	**0.60*** (0.03)	0.53 (0.06)	0.41* (0.03)

Table 12: **Similar results with disjoint judges only.** Per-sample win rates on the same MBPP+ comparisons as in the main results, but excluding votes from any judge that is also one of the compared models. Trends remain mostly similar, with the main difference being a lower win rate for the Advanced persona (0.40), suggesting limited sensitivity to the judge set. * denotes statistical significance on a two-sided binomial test.

Model Pair	Prompt Type	Win-rate (Tie-rate)			
		Beginner	Intermediate	AI Researcher	Advanced
GPT-5.1	Original	0.02* (0.93)	0.02* (0.93)	0.03* (0.93)	0.04 (0.93)
vs.	Personalized	**0.15*** (0.84)	**0.20*** (0.73)	**0.31*** (0.68)	**0.14*** (0.84)
GPT-4o	Control	0.02* (0.94)	0.02* (0.94)	**0.04*** (0.94)	**0.04*** (0.94)
GPT-5.1	Original	0.02* (0.94)	0.01* (0.94)	0.01* (0.94)	0.03 (0.94)
vs.	Personalized	**0.10*** (0.86)	0.15* (0.68)	0.18* (0.63)	**0.11*** (0.85)
GPT-OSS-20B	Control	0.02* (0.95)	0.01* (0.95)	0.01* (0.95)	**0.03*** (0.95)
Gemini-3-Pro	Original	0.13* (0.82)	0.14* (0.82)	0.14* (0.82)	0.14* (0.83)
vs.	Personalized	**0.23*** (0.75)	**0.30*** (0.66)	**0.41*** (0.58)	**0.26*** (0.68)
Gemma-3-4B	Control	0.13* (0.82)	0.14* (0.82)	0.14* (0.82)	0.14* (0.82)
Qwen3-32B	Original	0.03* (0.96)	0.04* (0.95)	0.03* (0.95)	0.03* (0.96)
vs.	Personalized	**0.10*** (0.80)	**0.33*** (0.53)	**0.25*** (0.60)	**0.10*** (0.83)
Qwen3-14B	Control	0.03* (0.94)	0.03* (0.94)	0.02* (0.94)	0.03* (0.94)

Table 13: **Similar results when jointly correct samples are treated as ties.** Per-sample win rates on the same MBPP+ comparisons as in the main results, but samples solved by both models are counted as ties. Personalized prompts still shift outcomes for the GPT and Qwen comparisons, while `Gemini-3-Pro` vs. `Gemma-3-4B` remains tie-dominated and shows a weaker effect. Therefore, this suggests that the observed preference shifts are mostly not driven by cases where both models are correct. * denotes statistical significance on a two-sided binomial test.

D.1 Data-Derived User Profiles

We use our Persona Creator that converts user information, such as public profiles, posts, or direct descriptions, into the structured profile format used by our pipeline. To provide a limited grounding check for our controlled archetypes, we apply it to two users represented in our in-the-wild corpus: a beginner with public posts in `r/learnpython` and the advanced coding YouTube channel *Fireship*. We extract only coding-related task and response preferences and manually review both profiles before use.

User Profile	Win-rate	
	Original	Personalized
Beginner Student	0.03	**0.77**
Advanced Developer	0.91	**0.58**
Reddit Beginner	0.04	**0.68**
YouTube Advanced	0.28	**0.53**

Table 14: **Data-derived profiles show similar directional preference shifts.** Win rates for `GPT-5.1` against `GPT-OSS-20B` on 25 MBPP+ samples, reported for the controlled archetypes and corresponding data-derived profiles. Personalized prompts increase `GPT-5.1`'s win rate for both beginner profiles and shift both advanced profiles in the same direction.

The Reddit-derived profile largely matches the controlled Beginner archetype (Table 14). The YouTube-derived profile reflects advanced coding expertise while also prioritizing simple, tutorial-style explanations for its audience. It therefore combines preferences associated with both advanced and beginner users.

We evaluate `GPT-5.1` against `GPT-OSS-20B` on 25 samples. For the Reddit-derived beginner profile, the `GPT-5.1` win rate increases from 0.04 on original prompts to 0.68 on personalized prompts, following the same direction as the controlled Beginner archetype, which increases from 0.03 to 0.77. For the YouTube-derived advanced profile, the win rate increases from 0.28 to 0.53. Its lower original win rate compared with the controlled Advanced archetype, which changes from 0.91 to 0.58, likely reflects its preference for simple tutorial explanations.

These results show similar directional personalization shifts for profiles derived from public user information. Although limited to two coarse profiles, this experiment provides initial evidence that the pipeline can be extended to real user data and motivates broader validation at both the population and individual levels.

D.2 Minimal-Personalization Control

Personalization often adds user context, constraints, and preferences, which naturally increases prompt length. To test whether length alone explains the observed preference shifts, we create minimal-personalization variants that add only one to three persona-relevant words to each prompt. For example, "Write a function..." becomes "Write an **efficient** function..." for the Advanced persona and "Write a **simple** function..." for the Beginner persona. These variants are also mostly shorter than the neutral paraphrase controls.

We evaluate `GPT-5.1` against `GPT-OSS-20B` under four prompt conditions. For the Beginner persona, the `GPT-5.1` win rate increases from 0.03 on original prompts and 0.06 on control prompts to 0.16 under minimal personalization, compared with 0.77 under full personalization. For the Advanced persona, it decreases from 0.91 on original prompts and 0.93 on control prompts to 0.84 with minimal personalization, approaching the full-personalization rate of 0.58.

For both personas, minimal personalization shifts preferences away from the original and control conditions and toward full personalization. The smaller shifts are expected because the intervention captures only a small part of each profile. Full personalization still combines several factors, including context, constraints, specificity, and requested response style. These results suggest that increased prompt length alone does not explain the main effect.

E LLM judge agreement analysis

To evaluate the reliability of the automated pairwise evaluation, we measure agreement between LLM judges on the final per-sample preference labels. Unless stated otherwise, we use `GPT-5.1`, `GPT-OSS-20B`, and `Qwen3-14B` as judges, with `GPT-5.1` omitted for Gemini and Qwen comparisons due to cost.

For each evaluated sample, each judge produces a final pairwise label for the preferred model (A WINS/B WINS/TIE) after resolving swapped-order comparisons as described in Section 6.2. We then compare judges on these final labels using two agreement measures: (1) raw agreement, the percentage of samples on which judges assign the same label, and (2) Fleiss's κ, which adjusts for chance agreement given the marginal label distribution (equivalent to Cohen's κ for two judges). To summarize agreement over a subset, we first split it into finer-grained conditions, compute judge-pair agreement within each condition, reduce each condition to a single mean agreement score, and then report the mean and standard deviation across conditions, weighted by the number of items in each condition. Since some subsets are label-imbalanced, for example when one model is preferred on most samples, κ can be lower even when raw agreement is relatively high.

E.1 LLM agreement results

Table 7 reports agreement across several pooled slices of the evaluation: by model pair, persona, prompt type, and judge pair. For each slice, we summarize condition-level agreement scores computed from the final preference labels. Values are reported as mean $\pm$ standard deviation across conditions, with each condition weighted by its number of items.

Overall, LLM judges show reasonably consistent preferences, with mean raw agreement of 78% $\pm$ 0.13 and Fleiss's $\kappa = 0.39 \pm 0.16$. Agreement is fairly stable across most model pairs, personas, and prompt types. By model pair, agreement is highest for `GPT-5.1` *vs.* `GPT-4o` and Gemini-3-Pro *vs.* Gemma-3-4B, and lowest for `Qwen3-32B` *vs.* `Qwen3-14B`, suggesting that this comparison is harder to judge consistently. Across personas, agreement is somewhat lower for the Researcher persona than for the others. Across prompt types, original, control, and personalized prompts yield broadly similar agreement, with only a small drop for personalized prompts. Finally, agreement is also similar across judge pairs, indicating that no single judge pair is driving the overall pattern.

These results support the use of LLM judges for subjective pairwise comparisons in our pipeline and show that agreement varies across subsets, with lower levels in some harder settings.

F Human validation of automated judgments

To validate whether our automated LLM-based pairwise judgments align with human assessments, we conducted a human preference annotation study. Annotators were presented with pairs of coding-assistant responses and asked to judge which response better fit a given user persona, both overall and across multiple quality dimensions. The study was designed to directly validate the persona-conditioned pairwise evaluation used in our pipeline.

F.1 Study setting

The study was grounded in the same evaluation framework used in the main experiments. We considered two prompt types: *original prompts*, taken directly from coding benchmarks, and *personalized prompts*, which rewrite the same tasks to reflect a specific user persona. We used two personas throughout the study: a *Novice User*, who values clarity and detailed explanations, and an *Advanced Developer*, who prefers concise, technical, high-signal responses. Each item compared the outputs of one of two model pairs: GPT-4o vs. GPT-5.1, and Gemini-3-Pro-Preview vs. Gemma-3-4B-IT. Model identities were hidden from annotators.

Pre-selection of items. Before the human study, candidate items were judged automatically by three LLM judges: GPT-5.1, GPT-OSS-20B, and Qwen3-14B. Judges compared model outputs from the perspective of a given persona and selected an overall winner, as well as winners for individual quality dimensions. Starting from a pool of 1,196 source items, filtering produced 667 items eligible for sampling. Filtering excluded items with overly long responses, unanimous overall ties, or insufficient judge agreement on the overall winner. Full rejection counts and filtering details are described in the study manifest files.

Sampling and design. The final study followed a 2 x 2 x 2 factorial design over persona, prompt type, and model pair. This yielded 8 condition cells. We sampled 32 unique items in total, using approximately balanced allocation across conditions. Two items were designated as calibration items and shown to all annotators; the remaining regular items were each assigned to exactly two annotators. Sampling used a fixed random seed.

F.2 Annotators and assignment

Six annotators participated in the study. Each annotator completed 12 items: 10 regular items and 2 shared calibration items. The resulting dataset contains 72 total annotation tasks, including 60 regular assignments and 12 calibration assignments. Regular-item assignment was balanced so that each annotator saw both personas, both model pairs, and both prompt types.

Annotation interface and questions. For each item, annotators were shown three elements: (1) a natural-language persona description, (2) the coding prompt, either original or personalized, and (3) two responses labeled A and B. Response order was randomized for half of the assignments to reduce position bias, and results were later mapped back to the canonical model order. Annotators selected *dimension-level preferences* between Response A / Tie / Responses B for seven evaluated dimensions: Clarity, Tone/Style Fit, Workflow Fit, Cognitive Load, Context Awareness, Persona Consistency, and Anthropomorphism. Finally, annotators reported overall response preferences and a confidence level of Low, Medium, or High. No free-text rationale was collected. Importantly, annotators were explicitly instructed that correctness was not the criterion of evaluation; instead, they were asked to judge which response better served the target persona.

F.3 Annotation difficulty and study scope

This annotation task proved demanding. It required annotators to understand code, compare often lengthy responses, and make judgments across seven dimensions from the perspective of a user persona that was not their own.

In post-task interviews, annotators described the task as "difficult" and "exhaustive." Several reported that, when comparisons contained especially long responses, they sometimes skimmed parts of the text and relied on heuristics to judge which answer was better. They also noted that it could be hard to assess whether a response truly satisfied persona-specific requests, especially when those requests involved technical preferences they did not fully share or understand, such as a particular complexity analysis or algorithmic explanation.

These difficulties were especially pronounced for personalized prompts, which tend to be longer because they add user-specific requirements and often elicit longer responses. This is not a flaw of the personalized prompts themselves: as our survey and in-the-wild analysis suggest, real vibe-testing often involves long, detailed, user-specific prompts and responses. Rather, it makes external validation by human annotators substantially harder. For this reason, we kept the study small, sampled relatively few personalized items, and omit their detailed results from the main text.

Overall, the study provides a focused human validation of the automated persona-conditioned judging setup used in the main experiments. It covers two personas, two prompt types, and two model pairs, and evaluates whether human preferences follow the same pairwise comparison framework used by the automated judges. Additional repro-

Prompt Type	Judge Pair Type	Agreement (%)	Cohen's κ	#Pairs	#Items
Original	LLM–LLM	90.9 ± 4.5	0.81 ± 0.10	3	66
	Human–Human	94.4 ± 15.0	0.80 ± 0.39	15	35
	Human–LLM	89.5 ± 15.6	0.78 ± 0.33	18	141
Personalized	LLM–LLM	100.0 ± 0.0	1.00 ± 0.00	3	30
	Human–Human	40.0 ± 43.1	-0.17 ± 0.58	15	24
	Human–LLM	50.0 ± 21.0	-0.06 ± 0.32	18	72

Table 15: Human judgment validation results across prompt types and compared grouped judge types. Agreement is reported as mean percentage agreement and Cohen's Kappa, with standard deviations across judge pairs. The reported Pairs counts judge pairs, and Items sums the per-pair overlap counts across those judge pairs.

ducibility details, including configuration files, item assignment manifests, and survey generation settings, are documented in the study materials.

F.4 Results

Table 15 summarizes agreement on the overall preference labels. On **original prompts**, agreement is high across all judge-pair types: LLM–LLM agreement is ($90.9\% \pm 4.5$) with ($\kappa = 0.81 \pm 0.10$), Human–Human agreement is ($94.4\% \pm 15.0$) with ($\kappa = 0.80 \pm 0.39$), and Human–LLM agreement is ($89.5\% \pm 15.6$) with ($\kappa = 0.78 \pm 0.33$). This indicates that, in the simpler original-prompt setting, automated persona-conditioned judgments align closely with human overall preferences. By contrast, agreement drops sharply on **personalized prompts**: while LLM–LLM agreement remains perfect on this small subset (note these samples were pre-sampled with high consensus), ($100.0\%, \kappa = 1.00$), Human–Human and Human–LLM agreement fall to ($40.0\% \pm 43.1$) and ($50.0\% \pm 21.0$), respectively, with near-zero or negative (κ). Given the small number of personalized items and the much greater annotation difficulty discussed above, we view these personalized-prompt results as noisy and inconclusive rather than as evidence against the automated setup.

Dimension-level agreement Tables 16 and 17 break the agreement down by evaluation dimension. On **original prompts**, the strongest agreement appears on *Tone/Style Fit*, where pooled agreement reaches ($92.3\% \pm 11.4$) ($\kappa = 0.81 \pm 0.23$), and on *Persona Consistency* after excluding ties, where pooled agreement reaches ($95.1\% \pm 18.6$) ($\kappa = 0.90 \pm 0.25$). Several other dimensions show moderate agreement once ties are excluded, including *Clarity* ($73.6\% \pm 25.1$), *Workflow Fit* ($75.0\% \pm 27.6$), and *Context Awareness* ($78.5\% \pm 32.7$). In contrast, *Anthropomorphism* is the least reliable dimension, with low pooled agreement ($38.8\% \pm 35.2$), ($\kappa = 0.05 \pm 0.13$). Overall, pooling all original-prompt sample-dimension decisions yields ($56.9\% \pm 14.2$) agreement and ($\kappa = 0.27 \pm 0.23$), rising to ($79.4\% \pm 14.7$) and ($\kappa = 0.51 \pm 0.33$) when ties are excluded. This suggests that much of the disagreement on original prompts comes from tie decisions rather than direct winner conflicts.

On **personalized prompts**, dimension-level agreement is substantially lower and much more variable. The pooled score across all dimensions is only ($34.9\% \pm 16.2$) ($\kappa = 0.03 \pm 0.21$), rising to ($61.3\% \pm 26.4$) ($\kappa = 0.09 \pm 0.39$) after excluding ties. LLM–LLM agreement remains noticeably higher than Human–Human or Human–LLM agreement, with pooled tie-excluded agreement of ($91.7\% \pm 2.1$) for LLM–LLM versus ($58.0\% \pm 34.0$) for Human–Human and ($58.9\% \pm 18.1$) for Human–LLM. Among dimensions, *Tone/Style Fit* and *Anthropomorphism* are relatively more stable after excluding ties, while dimensions such as *Workflow Fit*, *Context Awareness*, and *Persona Consistency* remain highly inconsistent. Together with the annotator feedback, these results suggest that personalized prompts are harder to validate externally: they often require reading longer responses, tracking more user-specific constraints, and judging from the perspective of a user whose preferences the annotator may not share.

Dimension	Pair Type	Agr. (%)	κ	Agr. excl. ties (%)	κ excl. ties
Clarity	Overall	54.4 ± 30.8	0.27 ± 0.32	73.6 ± 25.1	0.40 ± 0.42
	LLM–LLM	69.7 ± 6.9	0.43 ± 0.10	88.1 ± 6.1	0.70 ± 0.15
	Human–Human	48.7 ± 40.2	0.20 ± 0.34	75.0 ± 34.5	0.34 ± 0.47
	Human–LLM	56.6 ± 23.2	0.30 ± 0.33	70.3 ± 19.1	0.38 ± 0.42
Tone/Style Fit	Overall	92.3 ± 11.4	0.81 ± 0.23	97.6 ± 5.5	0.92 ± 0.16
	LLM–LLM	90.9 ± 4.5	0.78 ± 0.11	100.0 ± 0.0	1.00 ± 0.00
	Human–Human	97.8 ± 8.6	0.94 ± 0.18	100.0 ± 0.0	1.00 ± 0.00
	Human–LLM	88.0 ± 12.5	0.75 ± 0.25	95.2 ± 7.0	0.87 ± 0.19
Workflow Fit	Overall	54.1 ± 29.7	0.24 ± 0.35	75.0 ± 27.6	0.47 ± 0.47
	LLM–LLM	59.1 ± 16.4	0.32 ± 0.27	77.8 ± 19.2	0.53 ± 0.41
	Human–Human	55.7 ± 42.5	0.29 ± 0.45	75.0 ± 34.5	0.42 ± 0.49
	Human–LLM	52.0 ± 17.0	0.20 ± 0.30	74.5 ± 24.8	0.48 ± 0.50
Cognitive Load	Overall	50.0 ± 30.4	0.18 ± 0.35	65.9 ± 29.5	0.28 ± 0.51
	LLM–LLM	57.6 ± 2.6	0.33 ± 0.02	78.2 ± 5.6	0.55 ± 0.11
	Human–Human	42.9 ± 40.9	0.01 ± 0.37	58.5 ± 38.2	0.06 ± 0.45
	Human–LLM	54.7 ± 21.3	0.27 ± 0.34	68.4 ± 25.6	0.34 ± 0.55
Context Awareness	Overall	44.9 ± 36.2	0.22 ± 0.34	78.5 ± 32.7	0.45 ± 0.56
	LLM–LLM	39.4 ± 22.9	0.17 ± 0.22	72.1 ± 10.9	0.20 ± 0.35
	Human–Human	42.8 ± 46.4	0.19 ± 0.40	65.0 ± 47.4	0.17 ± 0.75
	Human–LLM	47.6 ± 29.1	0.24 ± 0.33	89.5 ± 16.7	0.70 ± 0.37
Persona Consistency	Overall	64.1 ± 36.9	0.43 ± 0.45	95.1 ± 18.6	0.90 ± 0.25
	LLM–LLM	53.0 ± 21.5	0.29 ± 0.26	94.7 ± 9.1	0.88 ± 0.20
	Human–Human	67.0 ± 46.1	0.43 ± 0.55	100.0 ± 0.0	1.00 ± 0.00
	Human–LLM	63.5 ± 31.1	0.46 ± 0.43	92.2 ± 24.4	0.86 ± 0.30
Anthropomorphism	Overall	38.8 ± 35.2	0.05 ± 0.13	59.7 ± 34.9	0.10 ± 0.17
	LLM–LLM	45.5 ± 25.3	0.18 ± 0.21	62.7 ± 26.3	0.27 ± 0.24
	Human–Human	26.7 ± 41.7	0.03 ± 0.10	56.2 ± 49.6	0.00 ± 0.00
	Human–LLM	47.8 ± 28.7	0.03 ± 0.13	60.7 ± 30.0	0.09 ± 0.16
Pooled	Overall	56.9 ± 14.2	0.27 ± 0.23	79.4 ± 14.7	0.51 ± 0.33
	LLM–LLM	59.3 ± 7.4	0.34 ± 0.09	83.1 ± 3.6	0.64 ± 0.08
	Human–Human	54.5 ± 17.3	0.16 ± 0.27	78.9 ± 19.9	0.39 ± 0.45
	Human–LLM	58.6 ± 12.5	0.34 ± 0.17	79.2 ± 10.5	0.57 ± 0.23

Table 16: **Dimension-level human judgment validation results on the original prompts.** Agreement is reported as mean percentage agreement and Cohen's κ, with standard deviations across judge pairs. Pooled counts treat each sample-dimension pair as one item. Excluding ties (`excl. tie`) removes items marked as ties by either judge, since ties do not affect which model wins.

Overall, the human validation results mostly support the automated judging setup in the original-prompt setting, where human and LLM judgments align closely at the overall level and reasonably well across several key dimensions. The personalized-prompt results are too small and inconsistent to validate the corresponding automated judgments, and annotation difficulty alone cannot rule out systematic judge bias. Stronger validation will require larger samples, annotators matched more closely to the target profiles, and annotation protocols that reduce response length and cognitive burden.

Dimension	Pair Type	Agr. (%)	κ	Agr. excl. ties (%)	κ excl. ties
Clarity	Overall	28.5 ± 32.7	-0.04 ± 0.35	47.6 ± 38.4	-0.13 ± 0.64
	LLM–LLM	83.3 ± 5.8	0.71 ± 0.08	96.7 ± 5.8	0.93 ± 0.12
	Human–Human	20.0 ± 36.8	-0.13 ± 0.32	35.7 ± 47.6	-0.20 ± 0.45
	Human–LLM	26.4 ± 21.8	-0.11 ± 0.22	43.3 ± 30.7	-0.33 ± 0.53
Tone/Style Fit	Overall	39.7 ± 35.3	0.08 ± 0.41	69.6 ± 38.7	0.28 ± 0.54
	LLM–LLM	76.7 ± 11.5	0.64 ± 0.18	95.2 ± 8.2	0.91 ± 0.16
	Human–Human	23.3 ± 37.2	-0.10 ± 0.32	43.8 ± 49.6	0.00 ± 0.00
	Human–LLM	47.2 ± 29.6	0.12 ± 0.41	78.3 ± 28.8	0.22 ± 0.62
Workflow Fit	Overall	34.4 ± 31.9	0.05 ± 0.41	52.6 ± 42.2	0.13 ± 0.61
	LLM–LLM	63.3 ± 5.8	0.45 ± 0.05	95.8 ± 7.2	0.92 ± 0.14
	Human–Human	16.7 ± 36.2	-0.14 ± 0.48	27.8 ± 44.1	-0.14 ± 0.38
	Human–LLM	44.4 ± 22.0	0.13 ± 0.31	57.9 ± 37.9	0.10 ± 0.63
Cognitive Load	Overall	34.2 ± 27.5	0.00 ± 0.30	62.9 ± 34.4	0.21 ± 0.46
	LLM–LLM	60.0 ± 10.0	0.39 ± 0.15	89.7 ± 9.0	0.79 ± 0.18
	Human–Human	36.7 ± 35.2	0.03 ± 0.21	66.7 ± 43.3	0.00 ± 0.00
	Human–LLM	27.8 ± 19.0	-0.08 ± 0.32	55.7 ± 30.2	0.15 ± 0.48
Context Awareness	Overall	35.1 ± 31.2	0.05 ± 0.37	60.6 ± 41.8	0.18 ± 0.55
	LLM–LLM	80.0 ± 0.0	0.66 ± 0.00	100.0 ± 0.0	1.00 ± 0.00
	Human–Human	20.0 ± 31.6	-0.10 ± 0.30	44.4 ± 52.7	-0.20 ± 0.45
	Human–LLM	40.3 ± 24.5	0.05 ± 0.34	62.3 ± 35.1	0.14 ± 0.46
Persona Consistency	Overall	38.6 ± 31.4	0.05 ± 0.31	69.6 ± 40.1	0.14 ± 0.69
	LLM–LLM	46.7 ± 15.3	0.20 ± 0.22	86.1 ± 12.7	0.72 ± 0.25
	Human–Human	36.7 ± 39.9	-0.08 ± 0.32	33.3 ± 57.7	-0.50 ± 0.71
	Human–LLM	38.9 ± 26.0	0.12 ± 0.30	73.8 ± 37.8	0.07 ± 0.67
Anthropomorphism	Overall	33.6 ± 32.9	0.01 ± 0.27	47.0 ± 36.8	-0.02 ± 0.33
	LLM–LLM	70.0 ± 10.0	0.53 ± 0.15	76.8 ± 9.3	0.53 ± 0.19
	Human–Human	40.0 ± 43.1	0.06 ± 0.13	68.8 ± 45.8	0.00 ± 0.00
	Human–LLM	22.2 ± 16.9	-0.11 ± 0.24	29.4 ± 23.8	-0.13 ± 0.27
Pooled	Overall	34.9 ± 16.2	0.03 ± 0.21	61.3 ± 26.4	0.09 ± 0.39
	LLM–LLM	68.6 ± 1.4	0.51 ± 0.03	91.7 ± 2.1	0.83 ± 0.04
	Human–Human	27.6 ± 14.3	-0.05 ± 0.14	58.0 ± 34.0	-0.16 ± 0.23
	Human–LLM	35.3 ± 11.1	0.01 ± 0.16	58.9 ± 18.1	0.10 ± 0.33

Table 17: **Dimension-level human judgment validation results on personalized prompts.** Agreement is reported as mean percentage agreement and Cohen's κ, with standard deviations across judge pairs. Pooled counts treat each sample-dimension pair as one item. Excluding ties (`excl. tie`) removes items marked as ties by either judge, since ties do not affect which model wins.

G Practitioner's Guide

The framework can be used as an evaluation scaffold for adapting existing tasks to a target user or community. The taxonomy provides a set of input dimensions for deciding what to test and output dimensions for deciding how responses should be judged. Practitioners can select the dimensions relevant to their setting, define their values from available user information, and implement each pipeline component with methods that fit their data and evaluation goals. The evaluation checklist shown in Table 18 summarizes the main decisions and validation steps involved.

Vibe-Testing Evaluation Checklist

Stage	Required Decision	Framework Elements	Validation and Reporting
1. Target definition	Define the target user, group, or community and the intended use of the evaluation.	User evidence, intended use, and evaluation scope.	Report the source of the user information, the target population, and the scope of the resulting profile.
2. Profile construction	Convert the available user information into explicit input and output preferences.	Input dimensions, output dimensions, and optional importance weights.	Leave unsupported preferences unspecified and manually review automatically generated profiles.
3. Task selection	Select tasks that reflect the target users' expected needs, goals, and workflows.	Task type, complexity, real-world context, constraints, and reference material.	Describe the domain and task coverage, including important settings that are not represented.
4. Prompt personalization	Apply profile-relevant changes to each task prompt.	Context, persona framing, specificity, constraints, underspecification, and requested response style.	Verify task preservation and report the prompt conditions, generation procedure, and controls.
5. Model generation	Run candidate models under a consistent comparison setup.	Prompt condition, model pair, number of generations, and comparison format.	Report model versions, system prompts, decoding settings, and generations per prompt.
6. Correctness evaluation	Apply objective or task-specific checks when reliable measures are available.	Correctness, constraint satisfaction, or task success.	Describe the tests, report their coverage, and identify relevant qualities they do not measure.
7. Subjective judging	Compare responses from the perspective of the target profile.	Selected output dimensions, profile preferences, and dimension-specific criteria.	Define each dimension clearly and mitigate position effects, judge bias, and model identity effects.
8. Aggregation	Combine dimension-level judgments into user-level or community-level summaries.	Dimension weights, tie handling, comparison rules, and profile aggregation.	Report the aggregation rule, tie rates, profile weights, and variation across users or groups.
9. Validation	Assess the reliability of the profiles, prompt transformations, judgments, and conclusions.	All pipeline components used in the evaluation.	Include appropriate human validation, agreement analysis, ablations, robustness checks, or manual review.

Table 18: **Evaluation checklist for applying the vibe-testing framework.** Each stage specifies a practical decision, the relevant framework elements, and the minimum validation or reporting needed to construct a clear and reproducible user-conditioned evaluation.

Defining the target user or community. A target profile can be constructed from a direct description, survey responses, public posts, interviews, support requests, or other user data. The source should provide evidence about the user's tasks, context, constraints, and response preferences. For example, a developer evaluating coding assistants for beginner Python students could use course surveys and forum questions to identify common needs, such as simple explanations, limited use of advanced syntax, and step-by-step debugging guidance. These observations can be mapped to input dimensions that shape the prompts and output dimensions such as clarity, cognitive load, and workflow fit. Multiple sources can be combined to represent a broader community, while individual profiles can be retained when the goal is to study variation between users. Profiles should be manually reviewed before use, especially when they are generated automatically from unstructured text.

Adapting an existing evaluation set. Practitioners can begin with an existing benchmark, internal task collection, or set of real user requests. For each task, they first identify which input dimensions are relevant to the target profile. They can then generate one or more prompt variants that add the corresponding context, constraints, or requested response style while preserving the underlying task. In the beginner Python example, a benchmark prompt asking for a function may be rewritten to request a simple implementation, avoid unfamiliar language features, and include a short explanation. Semantic-preservation checks, manual

review, or executable tests can help verify that the rewritten prompt still evaluates the intended capability. Neutral paraphrases can also be included to separate persona-specific effects from ordinary changes in wording.

Evaluating model responses. Models are run on the original and personalized task variants, and their responses are compared using the output dimensions selected for the target profile. When executable tests are available, correctness can be reported alongside the subjective judgments. Pairwise evaluation is often useful because it asks which response better serves the target user on each dimension. Practitioners can use human annotators, LLM judges, or a combination of both, depending on the task and available resources. The evaluation rubric should define each dimension clearly and explain how it applies to the target profile.

Inspecting and aggregating results. Results can first be inspected at the dimension level. This analysis can reveal, for example, that one model produces clearer explanations while another follows constraints more reliably. Dimension-level judgments can then be combined using importance weights from the user profile to produce an overall preference score. For a community-level analysis, practitioners can aggregate results across profiles, report the distribution of preferences, or group users with similar priorities. These summaries should preserve meaningful variation between users. Two users may assign different importance to clarity and concision, or may interpret workflow fit differently, leading to different model preferences. The resulting evaluation therefore supports user-conditioned comparisons and does not define a universal model ranking.

Open-ended tasks. The same procedure can be applied when automatic correctness metrics are unavailable. For tasks such as visual design, SVG generation, or dynamic webpage creation, practitioners can replace executable tests with task-specific rubrics, human review, constraint checks, or pairwise preference judgments. For example, a design team could compare generated webpages for accessibility, visual hierarchy, adherence to a brand style, and ease of modification. Some properties may still support automatic checks, such as valid HTML or accessibility rules, while subjective qualities can be evaluated by users or trained annotators. Reporting the dimensions separately helps distinguish disagreements about style and usability from failures to satisfy explicit requirements.

In practice, the framework is most useful when the target users, task collection, and evaluation dimensions are defined together. The taxonomy helps make these choices explicit, the pipeline applies them consistently, and the resulting analysis shows how model preferences depend on the users and settings being evaluated.

#	Source	Title
1	Youtube	Google's nano banana just killed Photoshop... let's run it
2	Youtube	GPT-5 is here... Can it win back programmers?
3	Youtube	Google's Genie model makes realistic worlds in realtime...
4	Youtube	Is Elon's Grok 3 the new AI king?
5	Youtube	I Tested Gemini 3 Pro (it did really well)
6	Youtube	Claude 4 vs GPT-5: Which AI is actually better?
7	Youtube	Claude vs GPT-5: The REAL Winner (Coding Test)
8	Youtube	Can AI Really Write Good Code? (GPT-5 vs Claude)
9	Youtube	How Well Do AI Models Actually Work? (Claude, GPT-5, Gemini)
10	Youtube	I Tried GPT-5 For A Week... Here's What Happened
11	Youtube	Claude 4 is here: first impressions and testing
12	Youtube	Gemini 3 Pro vs GPT-5: Real World Tests
13	Youtube	Best AI for writing in 2025? (Claude vs GPT)
14	Youtube	The AI model that finally feels human (testing)
15	Youtube	I compared AI chatbots for daily work (what surprised me)
16	Youtube	I Tested the BEST AI Coding Assistants (Shocking Results)
17	Youtube	GPT-5 vs Claude 4: 10 Real Use Cases
18	Youtube	Claude 4 Is Better Than GPT-5 (Here's Why)
19	Youtube	Gemini 3 Pro vs Claude 4: Which is better?
20	Youtube	I tested AI agents: what actually works

Table 19: Full list of in-the-wild sources used to construct the 40-report corpus (Part 1/2). URLs and additional source metadata are provided in the supplied CSV .

#	Source	Title
21	Reddit	GPT-5 vs Claude 4 for coding: which feels better?
22	Reddit	GPT-4o feels different, anyone else?
23	Reddit	What model do you prefer for writing and why?
24	Reddit	Benchmarks say X, but I keep using Y (reasons)
25	Reddit	Llama 3 vs Claude: vibes and reliability
26	Reddit	Which model has better tone for tutoring?
27	Reddit	Claude feels safer but more constrained, thoughts?
28	Reddit	What is the least frustrating model for daily work?
29	Reddit	Which model do you trust most and why?
30	Reddit	Why does this model "feel smarter" than the leaderboard?
31	Blog / News	Choosing an LLM: what benchmarks miss in practice
32	Blog / News	Real-world LLM eval: clarity, tone, and friction
33	Blog / News	I switched from GPT to Claude for my workflow
34	Blog / News	The best model is the one that fits your brain
35	Blog / News	AI leaderboards cannot predict user satisfaction
36	Blog / News	LLM adoption is driven by "feel," not scores
37	Blog / News	Why some models "feel smarter" despite lower benchmarks
38	Blog / News	Prompting for vibe: how I judge assistants
39	Blog / News	The model that feels most reliable for production use
40	Blog / News	DeepSeek vs. ChatGPT vs. Qwen 2.5: Here's the winner

Table 20: Full list of in-the-wild sources used to construct the 40-report corpus (Part 2/2). URLs and additional source metadata are provided in the supplied CSV.

Original Sample (MBPP+)

"Write a python function to check whether the two numbers differ at one bit position only or not."

Beginner Personalized Prompt (generated by GPT-5.1)

"I am `python_novice`, a student learning Python for introductory statistics using simple class-survey style examples, and I need to write a clear Python function that checks whether two integers differ at exactly one bit position only (no objects or advanced patterns, just a plain function I can run in a notebook). Please explain everything as if I only know basic Python, prefer readability over optimization, and structure the answer so it can be copied cell-by-cell into a Jupyter notebook, with short paragraphs starting with "Explanation:" before each code block describing what the code is about to do. After the main solution, include a brief recap of how each part of the code helps solve the task, connect the idea to simple stats-style situations (like thinking of binary flags in survey responses), and keep the required background limited to basic Python, not any special libraries. Also, after the explanation, suggest a few tiny experiments I can try (such as changing the input numbers or trying different bit patterns) and add a short reflection on what the results show about how the function works."

Intermediate Personalized Prompt (generated by GPT-5.1)

"I'm working on a tiny personal script that compares two integer IDs and needs a Python function `differ_by_one_bit(a, b)` that returns `True` only when the two numbers differ in exactly one bit position (and `False` otherwise), and it should run efficiently for lots of checks in a row, without worrying about micro-optimizations. I already have a broken stub:

```
def differ_by_one_bit(a, b):
# TODO: this is incomplete / may be wrong, fix or rewrite it
return None
```

I don't need basic syntax explained (like if/else or simple loops); instead, I want the work organized as (1) read/validate the two inputs; (2) determine whether they differ at exactly one bit; (3) return a boolean result, plus a short comment block above the function explaining in 2–3 sentences why the approach works and an 'Explanation' section after the code with brief Goal/Approach/Edge Cases notes. If anything about the requirement seems unclear, I want to pause and write down my clarifying questions first, and after my solution passes tests I'll add a short reflection on what I learned and one mistake I might have made without tests."

AI Researcher Personalized Prompt (generated by GPT-5.1)

"I want to add a small, reusable utility to my existing research repo: implement a Python function that, given two integer-like values, determines whether they differ in exactly one bit position, designed so it can be reused across different model-training pipelines and experiments. Please structure the code into small, pure functions with straightforward control flow, include explicit type hints for all public interfaces, and keep the design generic (not tied to any specific dataset or model) so it can be easily profiled, benchmarked, and extended by collaborators. In addition to the main boolean result, have the top-level function return a lightweight metadata dictionary (e.g., counts or intermediate values) to support downstream analysis and ablation studies."

Advanced Personalized Prompt (generated by GPT-5.1)

"I'm working on performance-sensitive code in a production service and need a clean, reliable helper for bit comparisons. Task: implement a Python function that determines whether two given integers differ at exactly one bit position, suitable for inclusion in our existing codebase and CI pipeline. Provide only the final implementation (with a simple, well-named API) plus a very brief note on the core idea and time/space complexity, written as you would for code review, with no extra commentary."

Figure 5: **Example of expertise-level personalized prompts generated by GPT-5.1 for an MBPP+ coding task.** Given a single original problem statement, the pipeline produces personalized prompts for the user profiles. Each of them reflects different assumptions about prior knowledge, desired explanation depth, and code style preferences.

Original Sample (MBPP+)

"Write a python function to check whether the two numbers differ at one bit position only or not."

Beginner Personalized Prompt (generated by Qwen3-32B)

"Hi there! I'm trying to learn how to compare numbers in Python using their binary form. Can you help me create a function that checks if two integers have exactly one different bit in their binary representation? I think this might involve some bitwise math, but I'm not sure how to count the differences. Let's make this simple to understand!

Requirements:

- The function should take two integers as input and return `True` if they differ at exactly one bit position, `False` otherwise
- Please explain each part of the code separately with plain language descriptions
- It would help if you mentioned how binary representations and bitwise operations relate to statistical concepts like binary differences
- Show step-by-step why each calculation is done that way

For example, 13 (`1101`) and 9 (`1001`) should return `True` because only the second bit differs, but 2 (`10`) and 4 (`100`) should return `False` since they have two differing bits. Can you guide me through building this logic from scratch?"

Intermediate Personalized Prompt (generated by Qwen3-32B)

"I'm building a utility function for my coding projects to check if two integers differ by exactly one bit position. First, I need a clear plan: explain your approach in plain English, then implement the solution in three stages—input validation, bitwise comparison logic, and final result formatting. Make sure to handle edge cases like maximum integer values, zero, and numbers with only one differing bit in higher positions. Write the function `differ_At_One_Bit_Pos(a, b)` that returns `True` when exactly one bit differs between the inputs, `False` otherwise. Verify your solution works for both small values like `(2, 3)` and large values like `(1048576, 2097152)` without performance issues."

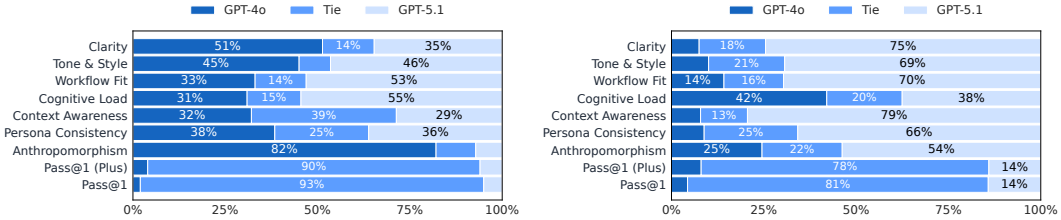
AI Researcher Personalized Prompt (generated by Qwen3-32B)

"I am developing a bit-level analysis module for a research project. My task is to create a Python function that determines whether two integers differ by exactly one bit position. The solution must be structured with modular components for bitwise operations and result validation. I need reusable functions with domain-specific names like `compute_bitwise_difference` and `validate_single_bit_variation` to ensure clarity. The implementation should include inline documentation for reproducibility in academic settings and configurable parameters for experimental testing. My code must pass standard test cases verifying single-bit differences across various integer pairs."

Advanced Personalized Prompt (generated by Qwen3-32B)

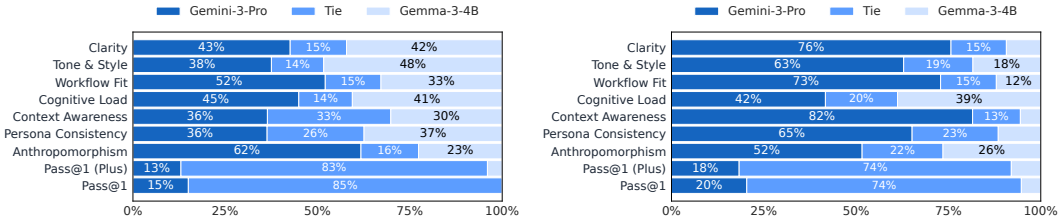
"As a low-level systems programmer optimizing for embedded devices, I need you to implement a memory-efficient Python function with $O(1)$ space complexity to determine if two integers exhibit Hamming distance of exactly 1. The solution must execute within 200ms latency constraints on an 8-core ARMv8 architecture with 1MB heap limit. Your implementation should leverage bitwise operations for optimal performance and include quantitative benchmarks comparing execution time and memory footprint against a naive implementation using bitstring conversion. Debugging must account for race conditions in parallel processing scenarios where multiple threads validate bit differences concurrently."

Figure 6: **Example of expertise-level personalized prompts generated by Qwen3-32B for an MBPP+ coding task.** Given a single original problem statement, the pipeline produces personalized prompts for the user profiles: *Beginner*, *Intermediate*, *AI Researcher*, and *Advanced*. Each reflects different assumptions about prior knowledge, desired explanation depth, and code style preferences.



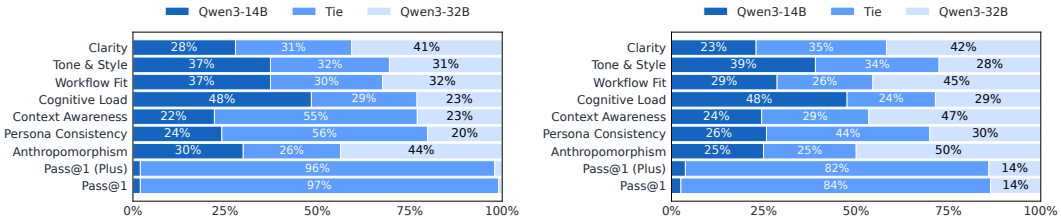
(a) **Original prompts** (b) **Personalized**

Figure 7: **Personalization changes model preferences GPT-5.1 vs. GPT-4o.** Head-to-head win rates for on MBPP+, broken down by dimensions. **Left:** original benchmark prompts. **Right:** persona-specific rewrites averaged over four personas. Several dimensions favor different models depending on the prompt form, showing that benchmark prompts can mask user-relevant differences beyond correctness.



(a) **Original prompts** (b) **Personalized**

Figure 8: **Personalization changes model preferences for Gemini-3-Pro vs. Gemma-3-4B.** Head-to-head win rates on MBPP+, broken down by dimensions. **Left:** original benchmark prompts. **Right:** persona-specific rewrites averaged over four personas. Several dimensions favor different models depending on the prompt form, showing that benchmark prompts can mask user-relevant differences beyond correctness.



(a) **Original prompts** (b) **Personalized**

Figure 9: **Personalization changes model preferences for Qwen3-32B vs. Qwen3-14B.** Head-to-head win rates on MBPP+, broken down by dimensions. **Left:** original benchmark prompts. **Right:** persona-specific rewrites averaged over four personas. Several dimensions favor different models depending on the prompt form, showing that benchmark prompts can mask user-relevant differences beyond correctness.

Survey Preamble Shown to Participants

Vibe-Testing AI Systems

Welcome to the Vibe-Testing AI Study!
Thank you for participating in our research on how people informally evaluate AI systems ("vibe-testing"). Your responses will help us understand real-world interaction patterns that are not captured by standard benchmarks.

Estimated Time: (∼)7–9 minutes
Voluntary Participation: Your participation is completely voluntary. You may stop at any time.
No Compensation: This survey does not include monetary compensation.
Privacy: Your responses will remain anonymous and used only for research purposes. We will not collect identifying information unless you choose to provide it at the end.
Stay Informed: If you would like to receive an email when the study results are published, please enter your email address at the end of this form. Your email, if provided, will be stored separately from your survey answers to protect your privacy.

By continuing, you acknowledge that you understand this information and consent to participate.

What we're asking you to do
This survey asks about how you interact with and evaluate AI systems when you use them. There are no right or wrong answers—we are interested in your honest experiences and opinions.

Vibe-Testing Definition (for this Survey)
For the purposes of this study, vibe-testing means informally evaluating an AI model through direct interaction, using your own prompts or tasks to judge how the model performs in practice. This includes noticing qualities such as: Usefulness for your workflow; Clarity and structure of responses; Tone or style; How it handles ambiguity; Whether it "feels" good to work with.
You may do this consciously (on purpose) or unconsciously (as part of how you explore models). We want to understand all such behaviors. Please answer candidly based on this definition and how you actually use AI models in your daily life.

Figure 10: Full preamble and informed-consent text shown to survey participants before the main questionnaire.

Question	Responses (%)
AI tool usage frequency. How often do you use AI tools (e.g., ChatGPT, Claude, Gemini, Copilot)?	**Daily:** 92.2% **Several times per week:** 7.8%
Primary use cases. What do you mainly use AI systems for? *(multi-select)*	**Coding / debugging:** 88.2% **Writing / communication:** 70.6% **Learning / tutoring:** 56.9% **Complex reasoning / problem-solving:** 37.3% **Productivity:** 35.3% **Creative tasks:** 33.3% **Entertainment:** 23.5% **Other (free-text):** 6.0%
Technical background. How would you describe your technical background?	**Technical (e.g., engineering, data science, CS background):** 47.1% **AI/ML expert:** 47.1% **Somewhat technical:** 5.9%

Table 21: Extended survey questions and results (Part A: **Usage and Background**). Percentages are computed over respondents who answered each question.

Prompt: Vibe-Test Extraction and Labeling

You are an NLP researcher analyzing transcripts to identify vibe-testing instances. A vibe-testing instance occurs when someone is:
(1) evaluating one or more AI models or systems, (2) describing a specific task they used to judge the models, and (3) using subjective, qualitative impressions rather than quantitative metrics.

Subjective language includes terms like: "vibes", "feels better/worse", "I prefer", "seems more", clarity, tone, workflow fit, reliability, consistency, verbosity, creativity, naturalness, hallucination patterns, "sounds like", "comes across as", or any other qualitative judgment about model behavior or output quality.

First, read the attached text.

Your task is to extract all vibe-testing instances from this text and return them as a JSON array.

`<scratchpad>`
Before creating your final JSON output, use this space to:

- identify potential vibe-testing instances,
- check each candidate against the criteria (evaluation + specific task + subjective language),
- note which subjective descriptors are used,
- verify that the instance describes an actual evaluation task, not just general discussion,
- exclude any mentions that are purely about benchmarks or quantitative metrics.

`</scratchpad>`

Each item in your JSON array must include these fields: `quote`, `task_type`, `models_mentioned`, `vibe_language`, `why_this_is_vibe_testing`, `benchmark_mention`, and `timestamp_range` (if available) or comment `metadata` (if available).

Important exclusions:

- Do NOT include instances where someone only discusses benchmark scores or quantitative metrics without subjective evaluation on a specific task.
- Do NOT include technical discussions about architecture or training without evaluation context.

Your output should consist only of the scratchpad followed by the final JSON array. The JSON should be properly formatted and valid.

Figure 11: **LLM prompt used for vibe-test extraction and labeling from YouTube transcripts and Reddit threads.** Minor formatting adjustments were made between YouTube and Reddit to reflect available metadata.

Prompt: Dimension-Based Re-Annotation

You are an expert NLP researcher annotating empirical examples for an ACL paper. Your job is to carefully annotate vibe-testing examples based only on the information explicitly present in the input JSON.

Critical rules:

- Do not infer facts that are not directly stated.
- Do not guess the user's intent, persona, or evaluation criteria unless clearly supported.
- If something is unclear or missing, mark it as `uncertain` or `not stated`.
- Do not add benchmark claims unless the quote explicitly mentions benchmarks.
- Prefer under-annotation over speculation.

Return a single JSON object following the provided schema, selecting vibe dimensions only from the closed set and justifying each assigned dimension using exact quoted language.

[Schema and closed dimension list as in the prompt shown in the main conversation.]

Figure 12: **LLM prompt used for dimension-based re-annotation.** Each vibe-test instance re-annotated with the fixed dimension set.

Prompt: Consistency Check and Gap Analysis

You will be provided with (1) the draft paper definitions and (2) a JSON of labeled vibe-testing instances. Your objective is to conduct a rigorous consistency check and gap analysis between the theoretical framework and the empirical data: verify that the definitions encompass the subjective language used, verify coverage of all distinct vibes, identify instances that do not fit the proposed dimensions, and identify paper dimensions that lack supporting evidence. Be critical and use direct quotes from the instances to support every critique.

Figure 13: **LLM prompt used for the final consistency check and gap analysis.** Provided together with the current definitions and the consolidated JSON as inputs.

Prompt: Persona Parsing

You are an expert user experience researcher. Your task is to analyze the following user description and generate a structured JSON profile based on it.

User Description:
"`{description}`"

Based on this description, create a JSON object that captures the user's persona, their needs regarding coding tasks, and their preferences for the output.

Your output MUST be a single, valid JSON object. Use the following structure and explanations as a guide.

JSON Template:
`{ . . . }`

Generate the JSON object now.

Figure 14: **Persona-parsing prompt.** Given a short natural-language user description, the LLM produces a structured JSON profile describing input and output preferences. The model must output a single JSON object and nothing else.

Prompt: Change Identification (Profile → Concrete Modifications)

You are an expert in personalizing programming tasks.

Task:
Given the following user profile, identify 2–3 concrete, distinct ways to modify a programming problem prompt for each field of the profile. The modifications should instruct how to adapt the prompt's phrasing, tone, and instructions to match the user's persona and preferences, without changing the core programming task. ...

User Profile:
—
{user_profile_json}
—

Output rules (MUST FOLLOW):
1. Output must be a single valid JSON object. No explanations, no comments, no markdown fences.
2. Root object has exactly one key: "changes_by_field".
   ...

Now, produce the JSON for all fields of the given user profile in exactly this format. Verify the output is a valid JSON object!

Figure 15: **Change-identification prompt.** To operationalize a persona profile into actionable prompt edits, the LLM proposes 2–3 concrete modification options for a fixed set of fields, while explicitly disallowing changes that alter the task itself. This stage outputs a single JSON object with a list of changes keyed by profile fields.

Prompt: Personalized Prompt Composition

You are an expert in rewriting programming tasks as a persona. You will be given an original programming problem prompt and a list of modifications to apply. Your task is to rewrite the prompt to incorporate all the specified changes while ensuring the underlying programming problem remains identical. ...

Original Prompt:
—
{original_prompt}
—

Modifications to apply:
—
{changes_description}
—

Instructions:
1. Carefully read the original prompt and the list of changes.
2. Rewrite the prompt to apply all changes cohesively.
3. DO NOT alter the core requirements of the programming task. The new prompt must lead to the same solution.
   ...
11. The length should not be longer than 2–4 short sentences.
12. Your output MUST only contain the prompt text and nothing else.

Figure 16: **Personalized prompt composition.** Given an original benchmark prompt and the selected modifications, the LLM generates a personalized version that preserves the underlying programming task. The prompt is written in the persona voice (first person), avoids explicit references to the profile schema, and is constrained to a short length.

Prompt: HumanEval+ Prefix Composition

You are an expert in rewriting programming tasks as a persona. ...
Your task is to write a prefix for the prompt to incorporate all the specified changes while ensuring the underlying programming problem remains identical. ...

Original Prompt:
—
{original_prompt}
—

Modifications to apply:
—
{changes_description}
—

Instructions:
...
10. The length should not be longer than 2–4 short sentences.
11. Your output MUST only contain the prefix text and nothing else.

Figure 17: **Prompt for HumanEval+ prefix composition.** For HumanEval+ style prompts that include code context and docstrings, only a short persona prefix is produced and concatenated to the original prompt, avoiding perturbation of code formatting while still injecting persona-relevant framing.

Prompt: Semantic-Preservation Verification

You are an expert in verifying programming problem statements. You will be given an original prompt and a modified version of it. Your task is to determine if the modified prompt preserves the original's core task and would result in the same ground-truth solution. ...

Original Prompt:
—
{original_prompt}
—

Modified Prompt:
—
{modified_prompt}
—

Decide:
- "same_end_goal": Is the fundamental task identical in both prompts?
- "same_ground_truth": Would the correct solution for the modified prompt also be correct for the original prompt?

...

Output MUST be a single valid JSON object, nothing else. Use exactly these keys: same_end_goal, same_ground_truth, reason_if_failed.

Figure 18: **Semantic-preservation verifier prompt.** To ensure personalized prompts remain faithful to the original benchmark intent, the verifier checks (i) whether the end goal is identical and (ii) whether the ground-truth solution set is preserved. The verifier returns two booleans and an error string if either check fails.

Prompt: Persona Creator

You will be writing a persona description for a programmer. A persona description is a brief first-person statement that captures the programmer's context (their role, experience level, work environment, etc.) and what they value when writing code (e.g., readability, performance, simplicity, etc.).

First, here are some example persona descriptions to use as style references:

```
<example_personas>
<1>
{example_persona_1}
</1>

<2>
{example_persona_2}
</2>
</example_personas>
```

Now, here are some Reddit posts written by the new user for whom you need to write a persona description:

```
<user_info>
{user_info}
</user_info>
```

Your task is to write a persona description for this new user that:

- Is written in first person (using "I", "my", etc.).
- Matches the style, tone, and approximate length of the example personas.
- Includes a brief description of the user's context (their role, background, work situation, or relevant circumstances).
- Describes what they value or prioritize in their code writing.
- Feels natural and authentic based on the information provided.

Study the example personas carefully to understand the typical structure and level of detail. Then craft a new persona description tailored to the new user's specific information.

Write your persona description inside `<persona>` tags.

Figure 19: **Persona creation.** Given example personas and public posts from a target user, the LLM generates a brief first-person persona description that captures the user's programming context and coding preferences. The generated description follows the style and approximate length of the provided examples.

Prompt: Short Minimal Personalization

You are an expert in subtly adapting programming task prompts to match a user persona. You will be given an original programming problem prompt and a brief user persona summary. Your task is to make ONLY minimal phrasing adjustments to the prompt that subtly reflect the user's background or style.

STRICT CONSTRAINTS:

- The modified prompt must NOT be longer than the original by more than 3–4 words.
- Do NOT add any new requirements, constraints, or instructions that were not in the original.
- Do NOT add explanations, examples, or clarifications that were not in the original.
- The modification should be limited to subtle word choice, phrasing, or contextual cues (e.g., adding "I need a function for my homework" for a student user).
- It must contain at least one user-specific detail for that persona from the user context or workflow mentioned in the possible modifications, without adding any new requirements, constraints, or instructions.
- The modified prompt should be human-like and natural, and not longer than the original beyond 3–4 words.
- Avoid phrasing such as "As a [PERSONA_DETAIL]...". Better phrasing is "I need to write...because I'm a [PERSONA_DETAIL]" or "I'm a [PERSONA_DETAIL], write...".
- The core programming task and its solution MUST remain identical.

User Persona Summary:

—

{persona_summary}

—

Possible modifications to apply (choose one):

—

{changes_description}

—

Original Prompt:

—

{original_prompt}

—

Instructions:

1. Read the original prompt and the persona summary.
2. Make only subtle phrasing changes (word substitutions, minor rewordings) that reflect the persona.
3. Keep the prompt nearly identical in length and structure to the original.
4. Do NOT add constraints, instructions, or any new content, including output format.
5. Your output MUST be a single valid JSON object with exactly one key "modified_prompt".
6. No extra keys, no trailing commas, no markdown fences, no commentary.

Correct Examples:

Novice user example
Original: "Write a function sum_list(nums) that returns the sum of a list of integers."
Persona: "A data science beginner learning Python"

{{"modified_prompt": "Help me write a function sum_list(nums) for my stat course, it should return the sum of a list of integers."}}

Advanced developer example
Original: "Write a function sum_list(nums) that returns the sum of a list of integers."
Persona: "An experienced developer"

{{"modified_prompt": "Write a highly optimized function sum_list(nums) that returns the sum of a list of integers."}}

Figure 20: **Short minimal personalization.** Given an original programming prompt, a user persona, and a set of possible modifications, the LLM makes a minimal persona-relevant phrasing change. The rewritten prompt is limited to an increase of at most three to four words and must preserve the original task, requirements, and solution.

Question	Responses (%)
Vibe-testing familiarity. Have you ever vibe-tested an AI model?	**Yes:** 82.4% **Not sure:** 13.7% **No:** 3.9%
Exploration frequency. When using a new AI model, how often do you experiment with prompts or tasks just to see how it behaves? (1 = never, 7 = very often)	**1:** 2.0% **2:** 4.1% **3:** 6.1% **4:** 8.2% **5:** 32.7% **6:** 22.4% **7:** 24.5%
Typical vibe-testing actions. Which of the following do you typically do when vibe-testing? *(multi-select)*	**Try tasks from my own workflow:** 73.5% **Compare outputs from different models:** 65.3% **Check tone, style, or personality:** 49.0% **Give vague or underspecified instructions (ambiguity stress-test):** 44.9% **Use a small set of personal "test prompts":** 38.8% **Re-run the same prompt (stability / consistency):** 38.8% **Other (free-text):** 6.0%
Vibe-testing domains. In which domains do you most often vibe-test AI tools? *(multi-select)*	**Coding / software development:** 65.3% **Writing / communication:** 51.0% **Reasoning / problem-solving:** 44.9% **Learning / tutoring:** 28.6% **Creative tasks:** 28.6% **Productivity:** 14.3% **Other (free-text):** 6.0%
Judgment criteria. When evaluating a model's responses, which aspects matter most to you? *(multi-select)*	**Correctness / accuracy:** 91.8% **Clarity and structure:** 59.2% **Fit for my personal workflow:** 40.8% **Tone / style:** 32.7% **Ability to resolve ambiguity:** 18.4% **Efficiency / conciseness:** 16.3% **Stability / consistency:** 20.4% **Other (free-text):** 14.0%

Table 22: Extended survey questions and results (Part B: **Vibe-testing Practices**). Percentages are computed over respondents who answered each question; multi-select questions report the percent selecting each option.

Question	Responses (%)
Semi-consistent prompt set. Do you have a semi-consistent set of prompts or tasks that you use when testing a new model? (Even just one)	**Improvised:** 51.1% **Partially consistent:** 34.7% **Consistent set:** 10.2%
Golden prompt existence. Do you have a "Golden Prompt" (a specific test you always use to quickly judge a model's quality)?	**Yes:** 18.4% **No:** 73.5% **Not sure:** 8.2%
Golden prompt content. If yes, please provide your "Golden Prompt" here.	*Open-ended; omitted for anonymization.*
Example prompt/task and what to look for. Please give one example of a prompt or task you might use when testing a model. Briefly say what you look for in the response.	*Open-ended; omitted for anonymization.*
Consent to publish prompts. May we include your prompts in our public report?	**Yes (anonymously):** 46.8% **Yes (with acknowledgement):** 4.3% **No:** 48.9%

Table 23: Extended survey questions and results (Part C: **Routines, Prompts, and Consent**). Percentages are computed over respondents who answered each question. Open-ended prompt fields were omitted for anonymization.

Question	Responses (%)
Benchmark awareness. Are you aware of the published performance scores (benchmarks) of the models you use?	**Yes:** 46.9% **Somewhat:** 42.9% **No:** 10.2%
Alignment with benchmarks. How closely do your personal impressions of models match their published benchmark results? (1 = not at all, 7 = very closely)	**1:** 0.0% **2:** 13.6% **3:** 9.1% **4:** 27.3% **5:** 36.4% **6:** 13.6% **7:** 0.0%
Benchmark mismatch. Have you ever found that a model *felt* significantly better or worse than its reported scores suggested?	**Yes:** 86.4% **Not sure:** 11.4% **No:** 2.3%
What benchmarks miss. Which aspects do you feel benchmarks fail to measure well? *(multi-select)*	**Fit for my workflow:** 61.4% **Style, tone, or personality:** 40.9% **Handling ambiguity / under-specification:** 34.1% **Stability and consistency:** 31.8% **Trustworthiness / safety:** 27.3% **Clarity and readability:** 27.3% **Other:** 13.6%
Value of vibe-testing. How valuable is vibe-testing for evaluating AI systems? (1 = not valuable, 7 = very valuable)	**1:** 0.0% **2:** 0.0% **3:** 2.0% **4:** 22.4% **5:** 38.8% **6:** 18.4% **7:** 18.4%
Interest in automation. If there were an automated tool that produced a personalized vibe-test for you, would you use it?	**Yes:** 32.7% **Maybe:** 51.0% **No:** 8.2% **Other (free-text):** 8.0%

Table 24: Extended survey questions and results (Part D: **Benchmarks, Gaps, Value, and Automation**). Percentages are computed over respondents who answered each question; multi-select questions report the percent selecting each option.