# FlowEval: Reference-based Evaluation of Generated User Interfaces

**Anonymous ACL submission**


## Abstract

While large language models (LLMs) and coding agents are often applied to user interface (UI) development, developers find it difficult to reliably assess their proficiency in visual and interaction design. Existing evaluations either rely on human experts, who can accurately assess usability by testing critical flows but are slow and costly, or on automated judges, which are scalable but less accurate and opaque. We present FlowEval, a reference-based framework that measures whether a generated UI supports realistic interaction flows by comparing navigation traces from real websites to traces from generated analogs using reference-based similarity metrics (e.g., dynamic time warping). In a study with expert UI evaluators, we show that reference-based metrics strongly correlate with human judgments, suggesting that they can provide scalable yet trustworthy evaluation for UI generation systems.


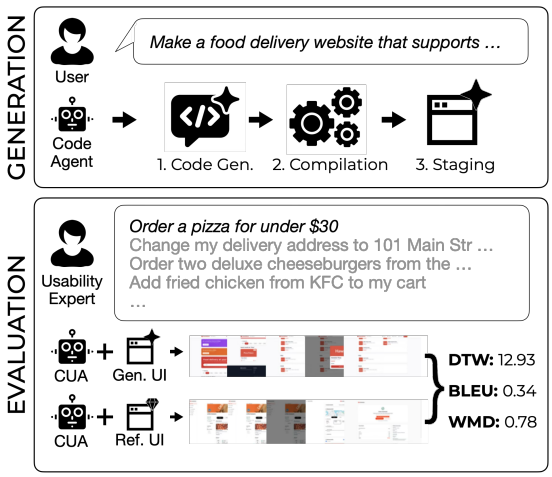


Figure 1: An overview of our evaluation approach. A coding agent generates an output, which is compared against a reference website. A computer use agent (CUA) produces and compares traces between generated and reference UIs.

## 1 Introduction

Recent advances in large language models (LLMs) and coding agents have made it possible to automatically generate complex user interfaces (UIs), but evaluating the quality of these interactive artifacts remains challenging. High-quality UIs must not only be visually appealing, but also functional and easy to use. Traditional approaches (Nielsen and Molich, 1990; Nielsen, 1994) to evaluating these factors rely on human expert review and testing, which can capture subtle design issues but are slow, costly, and difficult to scale.

One potential response is to use LLMs (or MLLMs) themselves as evaluators through prompting or fine-tuning on human responses, sometimes called "LLM-as-a-judge" (Zheng et al., 2023). Several such approaches have been specifically developed for assessing the quality of generated UI output (Zhang et al., 2025; Lin et al., 2025; Li et al., 2025a). However, such reference-free evaluators inherit the biases and failure modes of the underlying models and themselves require validation (Li et al., 2025b). Because of this, it can be difficult for domain experts to shape or trust these scores as an evaluation signal. Reference-based approaches, which have long been used in NLP and other areas of machine learning, offer a promising alternative. Compared to "reference-free" approaches, like automatic judges, they leverage ground-truth from known good examples, rather than relying solely on an evaluator model's internal notion of what a good output should be, which helps disentangle failures of the evaluator from genuine problems in the evaluated content. Domain experts can directly encode their subjective knowledge by curating reference designs and task sets for more targeted and bespoke evaluation, e.g., specific design criteria or adherence to a company's specific design language. Finally, metrics computed with references are also more interpretable and enable more targeted debugging, which is helpful for complex design and

interaction patterns.

We introduce FlowEval, a reference-based framework for evaluating interactive generated UIs. FlowEval uses a set of predefined interaction traces, or "flows," collected by performing tasks on a known high-quality UI and a generated analog to compute reference-based metrics. We describe an automated implementation that generates interaction traces by running a computer use agent (CUA) using validated tasks sourced from navigation benchmarks. We apply several types of reference-based metrics found in the NLP literature to compute similarity between traces collected from the reference and synthetic UIs. Finally, we conduct an arena evaluation where human experts provided 429 ratings of LLM-generated websites and compared it to our automated assessment method. Our results show that FlowEval's scores strongly correlate with human preferences and can provide more aligned rankings than a baseline MLLM judge approach used by previous work.

## 2 Related Work

Several benchmarks have been developed for assessing LLM code (Chen et al., 2021), which often include front-end languages (Peng et al., 2024). Many focus on tasks with references such as functional code correctness and "design-to-code" tasks (Si et al., 2025). However, it is difficult to automatically measure success for design quality.

Traditionally, this has relied on expert judgment (Nielsen, 1994), but this is difficult to scale. Prior research has explored encoding expertise into manually crafted objective functions (Todi et al., 2016; Oulasvirta et al., 2018), symbolic models (Ritter et al., 2000; Paternò, 2004), and guidelines (Nielsen and Molich, 1990; Shneiderman et al., 2016) for humans and machines to follow. Yet, it is hard to capture expert design knowledge in a form that is both comprehensive and broadly applicable.

Another approach is to engage a larger number of non-experts through crowdsourcing (Luther et al., 2015) or model "arenas" (Arcada Labs, 2025; Vichare et al., 2025). Because they cannot be run in a fully automated fashion, collected labels have, in turn, been used to improve automated approaches such as VLM judges for design aesthetics and (Duan et al., 2024; Wu et al., 2024) and interaction flows (Zhang et al., 2025; Lin et al., 2025; Li et al., 2025a). However, they can be less accurate due to high rater disagreement (Wu et al., 2024) and provide opaque scores.

Our work contributes to automated UI evaluation by retaining expert input through curated references, while using automated agents and metrics to enable scalable assessment.

## 3 FlowEval

We introduce FlowEval, a framework for evaluating UIs by generating interaction traces from validated tasks and computing similarity to high-quality reference flows. We describe our approach to i) generating interactive UIs that correspond to known references, ii) generating interaction traces using a CUA, and iii) applying three representative reference-based metrics to compute trace similarity.

**Source Data.** As a source of reference datasets and validated interaction tasks, we chose two CUA navigation benchmarks, although any expert-validated references and task flows can be used. We chose i) WebVoyager (He et al., 2024), which consists of 15 live websites and ii) REAL v2 (Garg et al., 2025), which provides 12 high-fidelity replicas for reproducible evaluation of stateful flows that are impractical to test on live sites (*e.g.,* authenticated or transaction-dependent interactions). WebVoyager contains a total of 643 tasks; but many of these tasks have a high degree of overlap, *e.g.,* multiple tasks that book a flight for different destinations. We distilled this into a smaller set by randomly sampling half of each website's tasks then manually removing repeated flows, resulting in 147 final tasks. We also manually filtered REAL's 121 tasks, which results in 116 final tasks.

### 3.1 UI Generation

We evaluated five LLMs, which included state-of-the-art proprietary and open-source LLMs at the time of our experiments: GPT 5 Codex (OpenAI, 2025), Gemini Pro 2.5 (Comanici et al., 2025), Claude Sonnet 4.5 (Anthropic, 2025), Qwen3Coder 30B, Qwen3Coder 480B (Qwen Team, 2025), GPT-OSS 20B, and GPT-OSS 120B (Agarwal et al., 2025). We used each model's official coding agent, which improved performance over manual one-shot model prompting through multi-turn execution and tool-calling: OpenAI Codex, Gemini CLI, Claude Code, and Qwen Code.

Each model was used to prompted to generate a UI in a "one-shot" manner using a React.js harness.

We modified a public prompt and harness used for UI benchmarking (Arcada Labs, 2025) to include a description of the website and a list of tasks that must be implemented (see Appendix). Each model generated 5 outputs for the same prompt, which led to a total of 945 outputs (27 reference sites, 7 models, 5 repeats).

### 3.2 Trace Generation

We used UI-TARS-1.5-7B (Qin et al., 2025), a vision-based CUA to capture UI states in reference and generated websites during task execution. We chose to use a vision-based agent, as opposed to one with access to the browser DOM, as we hypothesized it would more closely reflect real user perception and interaction with the UI.

At the beginning of each trace capture, a full-screen browser was launched a headless display server. The browser was pointed to the live website URL, or to a locally hosted build for generated outputs. The CUA was run using the officially provided library for UI-TARS, which operated the keyboard, mouse, and took screenshots for up to 50 steps. We used all default parameters except for temperature, which we increased from 0 to 0.7 for generating different traces.

### 3.3 Score Computation

A UI is scored by measuring the average similarity of its traces with the corresponding traces of a reference. To compute trace similarity, we selected three examples of reference-based scores found in NLP representing align-based, overlap-based, and distribution-based approaches.

**Featurization.** Each raw trace is converted into a sequence of $D$-dimensional embeddings, $x = (\mathbf{e}_1, \ldots, \mathbf{e}_T) \in \mathbb{R}^{T \times D}$. All traces were featurized using an off-the-shelf CLIP-style model finetuned on UIs (Wu et al., 2024) that maps each screenshot $i_t$ to a fixed-length vector $\mathbf{e}_t \in \mathbb{R}^D$.

**Aggregation.** To account for the variation between the three CUA runs, we computed the metrics for all nine possible pairings and select the best score.

#### 3.3.1 Alignment-based Scoring

For our experiments, we chose to use dynamic time warping (DTW) as an example of an alignment-based measurement (Sakoe and Chiba, 1978). DTW, an algorithm that first uses dynamic programming to find an optimal alignment through "warping" and computing a cost function from the optimal alignment path.

$$\mathrm{DTW}(x_{1:n}, y_{1:m}) = \min_{P \in \mathcal{P}} \sum_{(i,j) \in P} c(x_i, y_j) \quad (1)$$

Equation 1 describes DTW, where $\mathcal{P}$ refers to all possible "paths" between sequences $x$ and $y$, and $c(x, y)$ refers to the cosine distance between two elements $x_i$ and $y_j$.

To prevent unrealistic assignments, we applied Sakoe-Chiba bands of 10% of the reference trace length. If no valid warping was found, then the DTW was set to the length of the reference trace.

#### 3.3.2 Overlap-based Scoring

Overlap-based metrics compare sequences by the amount of sub-unit overlap (*e.g.,* n-grams). BLEU (Papineni et al., 2002) is a well-known example used for text, and we used eBLEU (El-Nokrashy and Kocmi, 2023), a generalization to continuous-valued sequences.

$$\mathrm{eBLEU}(x, y) = b \, \exp\Big(\sum_{k=1}^{K} w_k \log P_k(x, y)\Big) \quad (2)$$

Equation 2 describes the computation of eBLEU. $P_k(x, y)$ is the order-k "soft precision," defined as the average, over candidate k-grams in $x$, of the maximum clipped embedding-space similarity to any reference k-gram in $y$. Following the original paper, we considered k-grams up to length 4 and define $b$ as the length penalty for sequences with over a 50% difference in length.

#### 3.3.3 Distribution-based Scoring

Finally, we used Word Mover's Distance (WMD) as an example of a distribution-based measurement, which measures the the minimal optimal-transport cost to transform one sequence's distribution of features into the other (Kusner et al., 2015).

$$\mathrm{WMD}(x_{1:n}, y_{1:m}) = \min_{T \in \mathcal{U}} \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij} \, c(x_i, y_j) \quad (3)$$

Equation 3 describes the computation of WMD given two sequences $x$ and $y$, where $T$ is a transport plan, and $c(x, y)$ is the Euclidean distance between vectors $x$ and $y$.

## 4 Evaluation

We collected ratings of generated UIs from human experts and used them to measured alignment of our computed metrics and a MLLM-as-a-judge baseline. Ratings were collected in an arena-style evaluation (Chiang et al., 2024).

**Participants.** For this validation, the experts were two members of the research team who had PhD degrees in human-computer interaction (HCI) and were familiar with evaluation protocols such as usability testing and heuristic evaluation. The arena evaluation consisted of blind comparisons, reducing the chance of bias towards any specific condition. Despite the small number of evaluators, we found that our data were sufficient to reach statistical significance in determining the relative performance.

**Methodology.** We based our arena interface on existing ones for frontend development (Vichare et al., 2025; Arcada Labs, 2025). The arena page displayed a natural language description of a website but not the associated list of tasks, since we wanted raters to test tasks that they thought would be important given the description. Two websites from different models corresponding to the same randomly sampled prompt were displayed side-by-side in expandable `<iframe>` elements. Raters were instructed to interact with each website and test if it correctly implemented common functionality, then make a decision.

In total, we collected 429 unique pairs from human experts to serve as ground truth, which results in roughly 20 comparisons for each possible pairing of the 7 tested generator models (21 pairings). We computed Elo ratings for each code generator model following existing approaches (Chiang et al., 2024).

**Baseline.** For additional analysis, we implemented an automated judge (Li et al., 2025a) that prompted an MLLM to choose between a pair of candidates based the modified code and rendered UI screenshot. To run this judge, we used Claude Opus 4.5, the strongest available model for design tasks at the time of this experiment (Arcada Labs, 2025). We ran this judge on all human-rated pairs and computed Elo scores.

### 4.1 Results

Table 1 shows the Elo scores computed from human ratings, reference-based metrics, and MLLM judge. To measure the degree to which a metric aligned with human judgment, we adopted two approaches i) computing the Spearman rank correlation ($\rho$) and ii) computing the agreement rate.

| Model | Human | DTW | eBLEU | WMD | MLLM |
|---|---|---|---|---|---|
| Claude Sonnet 4.5 | 1342 (1) | 1210 (1) | 1098 (1) | 1146 (1) | 993 (5) |
| GPT 5.1 Codex | 928 (5) | 948 (5) | 969 (5) | 940 (5) | 1006 (4) |
| GPT-OSS 20B | 826 (6) | 909 (7) | 957 (7) | 915 (7) | 1020 (3) |
| GPT-OSS 120B | 776 (7) | 1021 (2) | 999 (3) | 928 (6) | 987 (6) |
| Gemini Pro 2.5 | 1048 (3) | 1008 (3) | 1024 (2) | 1033 (3) | 916 (7) |
| Qwen3Coder 30B | 972 (4) | 964 (4) | 998 (4) | 983 (4) | 1061 (1) |
| Qwen3Coder 480B | 1119 (2) | 944 (6) | 967 (6) | 1052 (2) | 1023 (2) |
| Spearman $\rho$ | N/A | 0.25 | 0.39 | **0.96** | 0.07 |
| Agreement rate | N/A | 68.1% | 61.1% | **73.0%** | 46.9% |

Table 1: Models Elo score (and rank) on the 429 unique pairs rated by human experts. WMD achieves the best Spearman correlation $\rho$ to human rankings.

Based on both approaches, WMD was the most effective, with near perfect rank correlation and 73.0% agreement rate. Other reference-based metrics aligned more closely to human labels than the baseline, and while not perfect, may offer other diagnostic utility, e.g., a DTW's alignment path reveals redundant steps in a flow. The MLLM performed the worst in both rank correlation and agreement rate. We hypothesized that this poor performance is due, in part, to the judge's heavy reliance on code to understand interactive functionality. Tested websites often contained thousands of lines of JavaScript code which required significant processing time and presented challenging "needle-in-a-haystack" problems for identifying implementation issues. We also note that despite using the authors' publicly released prompt (Li et al., 2025a), the judge performed considerably worse in our setup, likely because React.js embeds interaction logic in JavaScript and state, whereas the original authors' used pure-HTML sites.

Overall, these results provide evidence that reference-based metrics can improve alignment with human judgment and may address some weaknesses of automated judges.

## 5 Conclusion

We introduced FlowEval, a reference-based framework that evaluates generated UIs by comparing their interaction flows against those from reference websites. We show that reference-based metrics align strongly with expert judgments and outperform existing MLLM judges. FlowEval enables a form of automated yet reliable evaluation and can provide more interpretable feedback.

## Limitations

We discuss the limitations of FlowEval's intended scope and the implementation described in this paper.

**Scope.** UI quality is multi-dimensional and includes functional correctness, visual aesthetics, accessibility, performance, and overall user experience. FlowEval focuses primarily on task support, meaning whether a generated UI enables users to complete validated interaction flows. We view this as an underexplored axis for automated UI evaluation. As a result, FlowEval should be seen as complementary to prior approaches that emphasize compilation, layout quality, or visual appeal, and future work could combine reference-based flow metrics with these other signals to provide a more holistic assessment.

Reference-based evaluation also has inherent coverage limitations. High quality designs and experiences require creativity, and creative excellence may require designs that are intentionally novel or unconventional, which can make it difficult to identify an appropriate existing reference. Currently, we envision FlowEval as an automated approach to evaluating coding agents' implementation of commonly implemented UI flows and experiences.

FlowEval evaluates UIs in a fully automated setting, which requires generated websites to be produced without interactive feedback. In realistic development workflows, interfaces often go through multiple rounds of human iteration and refinement, which can substantially improve quality. Our results primarily characterize performance under a one-shot generation setting, not the quality achievable with human-in-the-loop iteration, although we feel that one-shot implementation performance is a reasonable proxy for more guided workflows.

**Implementation** Our implementation of the FlowEval evaluation framework is aimed to show feasibility of automated, reference-based flow evaluation. Although not tested, FlowEval could be extended to UIs outside of the web domain and include additional reference-based metrics beyond the three exemplars in our paper.

In our experiments, we sourced the tasks from agent navigation benchmarks, which may be biased toward publicly accessible flows. Many benchmarks under-represent stateful or restricted workflows, such as those requiring authentication, payments, or other gated interactions. Benchmark designers could curate a more challenging and representative set of tasks and expand coverage to a broader range of websites, workflows, and domains.

Our evaluation prompts are also more structured than many real-world requests to agents or generative models, which are often underspecified and do not enumerate explicit task requirements. Performance under FlowEval may therefore be viewed as an upper bound on models' ability to infer and support realistic tasks from ambiguous prompts. We hypothesize that performance in this constrained setting is predictive of performance in more naturalistic settings, but validating this relationship is an open problem.

Our experiments were also conducted with the best available APIs and agents available at the time of our experiments. The coding and navigation performance of LLMs are constantly changing and can affect the metrics computed from our method. A weak CUA may not be able to successfully execute a task, even if the website's implementation supports it. To some degree, we expect a CUA to maintain consistent performance across websites to mitigate this effect, e.g., failing the same task on both the reference and generated UI would still result in similar traces. However, we see an important avenue for future work in studying the robustness of our approach across multiple CUA agents.

**Generative AI Safety** Finally, FlowEval relies on generative models, including large language models and code-generating systems, which can produce harmful or unsafe outputs. This includes risks associated with unsafe code generation, policy-violating content, and agent behaviors that could lead to unintended actions during navigation. Practical deployments require careful sandboxing, content filtering, and safety guardrails.

## References


Sandhini Agarwal and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

akhaliq. 2025. Anycoder: Ai code generator with react frontend (hugging face space). Hugging Face Spaces demo. Accessed: 2026-01-02.

Anthropic. 2025. Claude sonnet 4.5 system card. Technical report, Anthropic.

Arcada Labs. 2025. Design arena: Crowdsourced benchmark for ai-generated design. https://designarena.ai/. Accessed: 2025-12-11.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 34 others. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 8359–8388. PMLR.

Gheorghe Comanici and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Peitong Duan, Chin-Yi Cheng, Gang Li, Bjoern Hartmann, and Yang Li. 2024. Uicrit: Enhancing automated design evaluation with a ui critique dataset. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA. Association for Computing Machinery.

Muhammad ElNokrashy and Tom Kocmi. 2023. ebleu: Unexpectedly good machine translation evaluation using simple word embeddings. In *Proceedings of the Eighth Conference on Machine Translation*, pages 744–748, Singapore. Association for Computational Linguistics.

Divyansh Garg, Shaun VanWeelden, Diego Caples, Andis Draguns, Nikil Ravi, Pranav Putta, Naman Garg, Tomas Abraham, Michael Lara, Federico Lopez, James Liu, Atharva Gundawar, Prannay Hebbar, Youngchul Joo, Jindong Gu, Charles London, Christian Schroeder de Witt, and Sumeet Motwani. 2025. Real: Benchmarking autonomous agents on deterministic simulations of real websites. *arXiv preprint arXiv:2504.11543*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *ICML*, pages 957–966. PMLR.

Chunyang Li, Yilun Zheng, Xinting Huang, Tianqing Fang, Jiahao Xu, Yangqiu Song, Lihui Chen, and Han Hu. 2025a. Webdevjudge: Evaluating (m)llms as critiques for web development quality. *Preprint*, arXiv:2510.18560.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, and 1 others. 2025b. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791.

Kevin Qinghong Lin, Siyuan Hu, Linjie Li, Zhengyuan Yang, Lijuan Wang, Philip Torr, and Mike Zheng Shou. 2025. Computer-use agents as judges for generative user interface. *Preprint*, arXiv:2511.15567.

Kurt Luther, Jari-Lee Tolentino, Wei Wu, Amy Pavel, Brian P. Bailey, Maneesh Agrawala, Björn Hartmann, and Steven P. Dow. 2015. Structuring, aggregating, and evaluating crowdsourced design critique. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 473–485. ACM.

Jakob Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann, San Francisco, CA.

Jakob Nielsen and Rolf Molich. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 249–256. ACM.

OpenAI. 2025. Addendum to gpt-5 system card: Gpt-5-codex. Technical report, OpenAI.

Antti Oulasvirta, Samuli De Pascale, Janin Koch, Thomas Langerak, Jussi Jokinen, Kashyap Todi, Markku Laine, Manoj Kristhombuge, Yuxi Zhu, Aliaksei Miniukovich, Gregorio Palmas, and Tino Weinkauf. 2018. Aalto interface metrics (aim): A service and codebase for computational gui evaluation. In *Adjunct Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18 Adjunct, pages 16–19, New York, NY, USA. Association for Computing Machinery.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Fabio Paternò. 2004. Concurtasktrees: an engineered notation for task models. *The handbook of task analysis for human-computer interaction*, pages 483–503.

Qiwei Peng, Yekun Chai, and Xuhong Li. 2024. HumanEval-XL: A multilingual code generation benchmark for cross-lingual natural language generalization. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8383–8394, Torino, Italia. ELRA and ICCL.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Qwen Team. 2025. Qwen3-coder: Agentic coding in the world.

Frank E Ritter, Gordon D Baxter, Gary Jones, and Richard M Young. 2000. Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(2):141–173.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. 2016. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6 edition. Pearson.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2Code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974, Albuquerque, New Mexico. Association for Computational Linguistics.

Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and explore with a layout optimiser. In *Proceedings of the 2016 ACM conference on designing interactive systems*, pages 543–555.

Aryan Vichare, Anastasios N. Angelopoulos, Wei-Lin Chiang, Kelly Tang, and Luca Manolache. 2025. Webdev arena: A live llm leaderboard for web app development.

Jason Wu, Yi-Hao Peng, Xin Yue Amanda Li, Amanda Swearngin, Jeffrey P Bigham, and Jeffrey Nichols. 2024. Uiclip: a data-driven model for assessing user interface design. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–16.

Chenchen Zhang, Yuhang Li, Can Xu, Jiaheng Liu, Ao Liu, Changzhi Zhou, Ken Deng, Dengpeng Wu, Guanhua Huang, Kejiao Li, Qi Yi, Ruibin Xiong, Shihui Hu, Yue Zhang, Yuhao Jiang, Zenan Xu, Yuanxing Zhang, Wiggin Zhou, Chayse Zhou, and Fengzong Lian. 2025. Artifactsbench: Bridging the visual-interactive gap in llm code generation evaluation. *Preprint*, arXiv:2507.04952. ArXiv v2 (last revised 29 Sep 2025).

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS 2023 Datasets and Benchmarks Track*.

## A UI Generation Prompt

We generated React.js websites with several coding agents using a one-shot prompt. Our one-shot generation prompt was based on pubicly available prompts used for UI generation and benchmarking (Arcada Labs, 2025; akhaliq, 2025). We modified existing prompts specifically for generating code that could easily be staged in our React.js harness (*e.g.,* only modifying files in a certain directory and using only certain libraries).

```
You are an expert on frontend design,
you will always respond to web design
tasks.
Your task is to create a website
according to the user's request using
the React framework.
When implementing the website, you
should follow these rules:
[Implementation Rules]
1. You should use React by default.
2. If the user requires a library that
is not installed in current react
environment, please use HTML and tell
the user the reason.
3. Your task is to create a website
using React JSX, where the entire
implementation is contained within the
src/ directory. src/App.jsx is the main
view and it should should export a
component called App. Besides the new
files you create in the src/ directory,
you must not change or edit any other
files.

## Common Design Principles

### General Design Guidelines:
- Create a stunning, contemporary, and
highly functional website based on the
user's request
- Implement a cohesive design language
throughout the entire website/
application
- Choose a carefully selected,
harmonious color palette that enhances
the overall aesthetic
- Create a clear visual hierarchy with
proper typography to improve readability
- Incorporate subtle animations and
transitions to add polish and improve
user experience
- Ensure proper spacing and alignment
using appropriate layout techniques
- Implement responsive design principles
 to ensure the website looks great on
all device sizes
- Use modern UI patterns like cards,
gradients, and subtle shadows to add
depth and visual interest
- Incorporate whitespace effectively to
create a clean, uncluttered design
- For images, use placeholder images
from services like https://placehold.co/
```

```
## React Design Guidelines

### Implementation Requirements:
- You MUST export an **`App`** component
 as default export in src/App.jsx
- DO NOT include any external libraries,
 frameworks, or dependencies outside of
what is already installed
- Utilize TailwindCSS for styling,
focusing on creating a visually
appealing and responsive layout
- Do not define any new SVGs - you
should use icons from FontAwesome
instead from the packages: @fortawesome/
fontawesome-svg-core @fortawesome/free-
solid-svg-icons @fortawesome/react-
fontawesome. You can assume that these
packages are installed even though you
cannot see them in a package.json in the
 same directory.
- Avoid using arbitrary values (e.g., `h
-[600px]`). Stick to Tailwind's
predefined classes for consistency
- Use static data instead of making HTTP
 requests or API calls to external
services. Define all data classes and
placeholder data structures in a
location so that it can be referenced
when creating components. Since a lot of
 the website's functionality may depend
on this static data, you should think
carefully about what should be inside of
 it, define it early, and hook it up to
various parts of the app.
- Utilize Tailwind's typography classes
to create a clear visual hierarchy and
improve readability
- Ensure proper spacing and alignment
using Tailwind's margin, padding, and
flexbox/grid classes
- All functionality, such as buttons,
links, search functionality, and forms
should be implemented.
- Do not treat this as the starting
point for an app - it should be the
final complete UI.
- Remember to include alt text for all
images.
- There should absolutely NEVER be any
dead links or buttons that aren't
implemented.

Create a website with the following
description: <App Name>: <App
Description>. The website must support
the following tasks with the entire
flows realistically mocked:

- Task 1
- Task 2
- Task n

After implementing those tasks, do the
following:
- Imagine what other functionality the
website is missing. Be as comprehensive
as possible. Think of all the possible
reasons why a user might visit this
website. Add that to your todo list
- Implement mocked flows for those
functionality.
- Make sure all remaining buttons and
interactive elements on the website
should be mocked up and respond to
interactions realisitically.

After you are done, review the code you
have generated and ensure it meets all
of the requirements. Do not run the
server or install additional packages.
Never execute the `npm` command.
```

## B Arena Labeling Interface

Figure 2 shows the rating interface used by human raters during our experiment. It is based on the design of existing interface used for rating UIs (Vichare et al., 2025; Arcada Labs, 2025).

## C MLLM Judge Prompt

Following previous work (Li et al., 2025a), we employed an MLLM judge in pair mode. For each comparison, the assignment of candidates to Model A and Model B was randomized to mitigate positional bias.

Each request included: i) the user query (app name + description), ii) Code A and Code B (full contents of files that were *new or modified* relative to the boilerplate), and iii) two screenshots labeled "Initial State A" and "Initial State B."

The judge prompt followed the publicly released WebDevJudge Likert rubric (Li et al., 2025a), with output required as JSON scores for A/B per sub-criterion:

```
{
  "1.1": {"A": 1-5, "B": 1-5},
  ...,
  "4.3": {"A": 1-5, "B": 1-5}
}
```

We then converted scores to a pairwise decision by summing criterion scores and choosing the model with the higher score.

## D Raw Metric Values

Our main experiment computed the reference-based metrics for each comparison pair and used that to create a binary label (Table 1). These binary labels were used to calculate an Elo score that was comparable to our human ranking data. Table 2 uses a difference approach were the raw metric values are aggregated (median) across all model outputs, which is sorted to generate a ranking. We found that this approach did not perform as well as

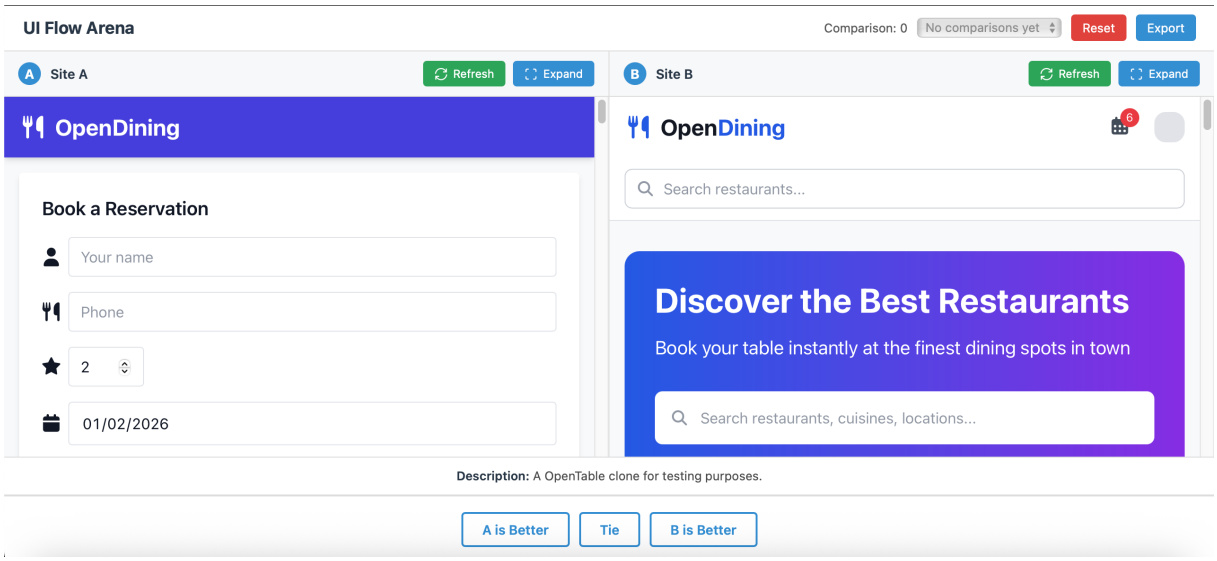


Figure 2: A screenshot of our arena interface used by human raters in our experiment.

| Model | DTW ↓ | eBLEU ↓ | WMD ↓ |
|---|---|---|---|
| Claude Sonnet 4.5 | 9.550 (2) | **0.226 (1)** | **0.603 (1)** |
| Qwen3Coder 480B | 15.014 (7) | 0.355 (4) | 0.640 (2) |
| Gemini Pro 2.5 | **8.636 (1)** | 0.319 (2) | 0.668 (4) |
| Qwen3Coder 30B | 14.902 (6) | 0.383 (7) | 0.663 (3) |
| GPT 5.1 Codex | 13.788 (5) | 0.371 (5) | 0.691 (7) |
| GPT-OSS 20B | 13.405 (4) | 0.381 (6) | 0.687 (6) |
| GPT-OSS 120B | 11.542 (3) | 0.345 (3) | 0.686 (5) |
| Spearman $\rho$ vs GT rank | 0.04 | 0.46 | **0.82** |

Table 2: Reference metric medians (rank in parentheses) on the comparison pairs rated in our experiments. Spearman $\rho$ is computed against rankings derived from human Elo score. WMD remains the best performing metric.

the Elo method, but we include the values here for reference.