WebVR: Benchmarking Multimodal LLMs for WebPage Recreation from Videos via Human-Aligned Visual Rubrics

Anonymous ACL submission

Abstract

Existing web-generation benchmarks rely on text prompts or static screenshots as input. However, videos naturally convey richer signals such as interaction flow, transition timing, and motion continuity, which are essential for faithful webpage recreation. Despite this potential, video-conditioned webpage generation remains largely unexplored, with no dedicated benchmark for this task. To fill this gap, we introduce WebVR, a benchmark that evaluates whether MLLMs can faithfully recreate webpages from demonstration videos. WebVR contains 175 webpages across diverse categories, all constructed through a controlled synthesis pipeline rather than web crawling, ensuring varied and realistic demonstrations without overlap with existing online pages. We also design a fine-grained, human-aligned visual rubric that evaluates the generated webpages across multiple dimensions. Experiments on 19 models reveal substantial gaps in recreating fine-grained style and motion quality, while the rubric-based automatic evaluation achieves 96% agreement with human preferences. We release the dataset, evaluation toolkit, and baseline results to support future research on video-to-webpage generation.

1 Introduction

MLLMs (Lin et al., 2024; Bai et al., 2025; Huang et al., 2026) are rapidly advancing toward end-to-end generation of executable artifacts from visual inputs. In web development, designers frequently provide screen recordings rather than fully specified design documents to communicate layout, interaction flow, and animation timing. These recordings naturally encode both static appearance and dynamic interactions, making them rich references for front-end implementation. Recent advances in MLLMs have made it feasible to convert such demonstration videos directly into executable code, substantially reducing development effort. However, the actual capabilities of current MLLMs in faithfully recreating webpages from video remain largely underexplored.

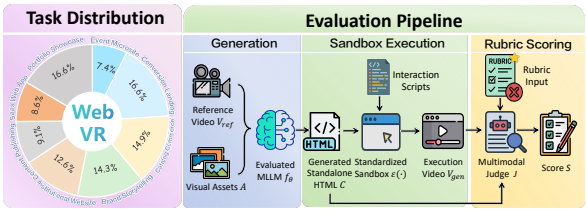


Figure 1: Overview of the WebVR benchmark. The left panel illustrates the task distribution across various webpage categories, while the right panel details the automated evaluation pipeline, which executes generated code in a standardized sandbox and scores it against human-aligned visual rubrics.

Existing benchmarks have advanced webpage evaluation but remain limited in two aspects. First, they primarily evaluate generation from text prompts or static screenshots (Xu et al., 2025; Lu et al., 2025; Sun et al., 2025; Laurençon et al., 2024; Yun et al., 2024), leaving dynamic behaviors such as transitions and animations unevaluated. Even recent efforts that approximate dynamics via staged screenshots (Zhang et al., 2025) provide only sparse temporal evidence, failing to capture the fluid motion essential to modern web interfaces. Second, existing evaluation protocols rely on coarse-grained or structure-oriented criteria, lacking a principled rubric for assessing fine-grained visual and interaction fidelity.

To fill these gaps, we introduce **WebVR**, the first benchmark for video-to-webpage generation. WebVR evaluates whether MLLMs can transform demonstration videos into fully functional front-end implementations with high visual and interactive fidelity. It contains 175 webpages across diverse categories such as e-commerce, portfolio, landing page, entertainment, and education. All webpages are constructed through a controlled synthesis pipeline rather than web crawling, ensuring

varied and realistic demonstrations without overlap with existing online pages. To support fair evaluation, WebVR provides visual assets alongside each task, so that models focus on faithfully reconstructing layout and interactions rather than sourcing suitable images. We also design a fine-grained, human-aligned visual rubric that evaluates the generated webpages across multiple dimensions including layout structure, color and typography, component completeness, animation quality, and interaction correctness. The generated code is rendered and recorded in a standardized sandbox, and an MLLM-based judge scores the execution video against the reference video under this rubric, enabling scalable, reproducible, and interpretable assessment. An overview of WebVR is shown in Fig. 1. Experiments on 19 models reveal substantial gaps in recreating fine-grained style and motion quality, while the rubric-based automatic evaluation achieves 96% agreement with human preferences.

The contributions of this work are as follows:

- **Video-to-Webpage Benchmark.** We introduce **WebVR**, the first benchmark for video-to-webpage generation. It covers 175 visually rich webpages across diverse categories, constructed through a controlled synthesis pipeline with paired visual assets for fair evaluation.
- **Video-Based Evaluation Protocol.** We propose an evaluation protocol that renders the generated front-end code in a standardized sandbox and records execution videos, enabling reproducible assessment of both static appearance and dynamic interactions beyond static snapshots.
- **Human-Aligned Visual Rubric.** We design a fine-grained, human-aligned visual rubric that guides an MLLM-based judge (Chen et al., 2024) to score generated webpages across multiple dimensions, producing interpretable, dimension-level feedback and reliable model rankings that achieve 96% agreement with human preferences.

2 Related Work

Webpage Generation. Recent progress in LLMs (Naveed et al., 2025; Dai et al., 2025) has enabled translating natural language specifications into executable web code. Several benchmarks evaluate this capability from different input modalities. Text-based benchmarks such as ArtifactsBench (Zhang et al., 2025) and Interaction2Code (Xiao et al., 2025) approximate dynamic behaviors through staged screenshots, but this sparse sampling fails to capture fine-grained motion patterns, animation timing, and continuous transitions. WebRRSBench (Liu et al., 2025) broadens evaluation to reasoning, robustness, and safety, yet does not address dynamic visual fidelity. Image-based benchmarks including WebSight (Laurençon et al., 2024), Web2Code (Yun et al., 2024), DesignBench (Xiao et al., 2026), and WebUIBench (Lin et al., 2025) advance visual fidelity by translating screenshots or UI slices into code. However, they remain grounded in static inputs and provide no supervision over temporal dynamics such as animation pacing and user-triggered state transitions. In contrast, video demonstrations inherently encode rich spatio-temporal signals including motion trajectories, scrolling behaviors, and hover feedback. To the best of our knowledge, no existing benchmark systematically evaluates the recovery of these fine-grained dynamic behaviors from video demonstrations, which is the focus of this work. We summarize the key differences between existing webpage benchmarks and our WebVR in Table 1.

MLLM as Judge. Traditional DOM or pixel-based metrics (Radford et al., 2021) often overlook high-level semantics and interaction flows. Recent work has explored using MLLMs as evaluators: WebDevJudge (Li et al., 2026a) and MLLM as a UI Judge (Luera et al., 2025; Li et al., 2026b) assess web development quality and human perception of UIs, respectively, while WebCoderBench (Liu et al., 2026) introduces 24 fine-grained metrics combining rule-based and LLM-as-a-judge paradigms. The emergence of LLM-as-a-Judge (Zheng et al., 2023), together with advances in multimodal models such as Qwen3-VL (Bai et al., 2025) and Kimi-K2.5 (Team et al., 2026), has made it feasible to assess visual fidelity and layout coherence directly from rendered webpages. However, prior studies report challenges including hallucination (Wang et al., 2024), prompt sensitivity, and limited awareness of engineering quality. WebVR addresses these by employing a fine-grained visual rubric aligned with human preferences, guiding the judge to produce reliable and interpretable scores.

Table 1: Feature Comparison of Existing Webpage Benchmarks (AP: Asset-Provided; GL: Granularity-Level)

Benchmark	Input Modality	Data Source	Evaluation Modality	AP	GL
Web2Code	Image	Synthetic	Image	×	Low
WebBench	Text	Human-Written	Code	×	Mid
WebUIBench	Image	Real-World	Code & Image	×	Mid
FullFront	Image	Synthetic	Code & Image	×	Mid
ArtifactsBench	Text	Synthetic	Code & Image	×	High
WebVR (Ours)	**Video**	**Synthetic**	**Code & Video**	**✓ (Fair)**	**High**

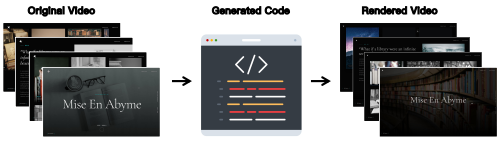


Figure 2: A case study illustrating the conversion from an original video into generated code and its rendered video output.

3 Method

3.1 Task Definition

We define the task of video-conditioned webpage recreation. Given a reference screen-recording video V_{ref} that demonstrates the target webpage's layout, visual style, and interactive behaviors, along with a set of visual assets $A = \{a_1, a_2, \ldots, a_m\}$ (e.g., images and icons), an MLLM f_θ is required to generate a standalone, executable HTML document:

$$C = f_\theta(V_{ref}, A) \tag{1}$$

Unlike screenshot-to-code tasks that focus on static visual cloning, this task requires the model to accurately reproduce both spatial layout and dynamic interactive behaviors demonstrated in the video.

Dynamic Execution and Rendering. To evaluate the true user-facing output, we introduce a standardized sandbox environment $\mathcal{E}(\cdot)$ that executes the generated HTML C under a fixed interaction script and records the resulting screen-capture video:

$$V_{gen} = \mathcal{E}(C) \tag{2}$$

This enables direct visual comparison between V_{gen} and V_{ref}, capturing both static appearance and dynamic state transitions. Fig.2 provides a visual case study of this end-to-end process, illustrating the conversion from the original reference video to the generated code and its rendered output.

Evaluation via Visual Rubrics. A perfect recreation would yield $V_{gen} \approx V_{ref}$. However, direct pixel-wise or frame-by-frame comparison is brittle for generative tasks, as it penalizes minor yet acceptable variations in animation easing or rendering differences. Instead, we decompose visual and interactive consistency into a set of fine-grained, atomic rubric criteria $R = \{r_1, r_2, \ldots, r_K\}$, uniquely defined for each reference video V_{ref}. Each criterion r_k is a binary verification condition targeting a single visual property, such as whether a specific element exists, whether its alignment is correct, or whether a hover animation triggers as expected. An MLLM-based judge J evaluates the generated output against the full rubric R in a single inference pass. The judge takes both the execution video V_{gen} and the generated HTML C as input, and predicts the binary satisfaction status for each criterion, producing $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K\}$ where $\hat{y}_k \in \{0, 1\}$. The overall recreation quality score is computed as:

$$S = \frac{1}{K} \sum_{k=1}^{K} \hat{y}_k \tag{3}$$

This formulation shifts webpage evaluation from coarse structural or textual metrics to a visually grounded, temporally aligned verification process.

3.2 Seed Data Preparation

A primary challenge in benchmarking MLLMs is data contamination, since these models have likely been exposed to most existing web content during pre-training. To ensure unseen evaluation instances, each WebVR instance is built through a three-stage synthesis pipeline (Fig. 3): (i) collecting real-world webpage demonstration videos, (ii) transcribing them into structured captions, and (iii) re-theming each caption into a fictional but structurally equivalent specification.

Video Collection. We collect high-resolution screen-recorded showcase videos from design-gallery platforms such as Landing Love, Godly, Lapa, etc., preserving fine-grained typography and motion cues.

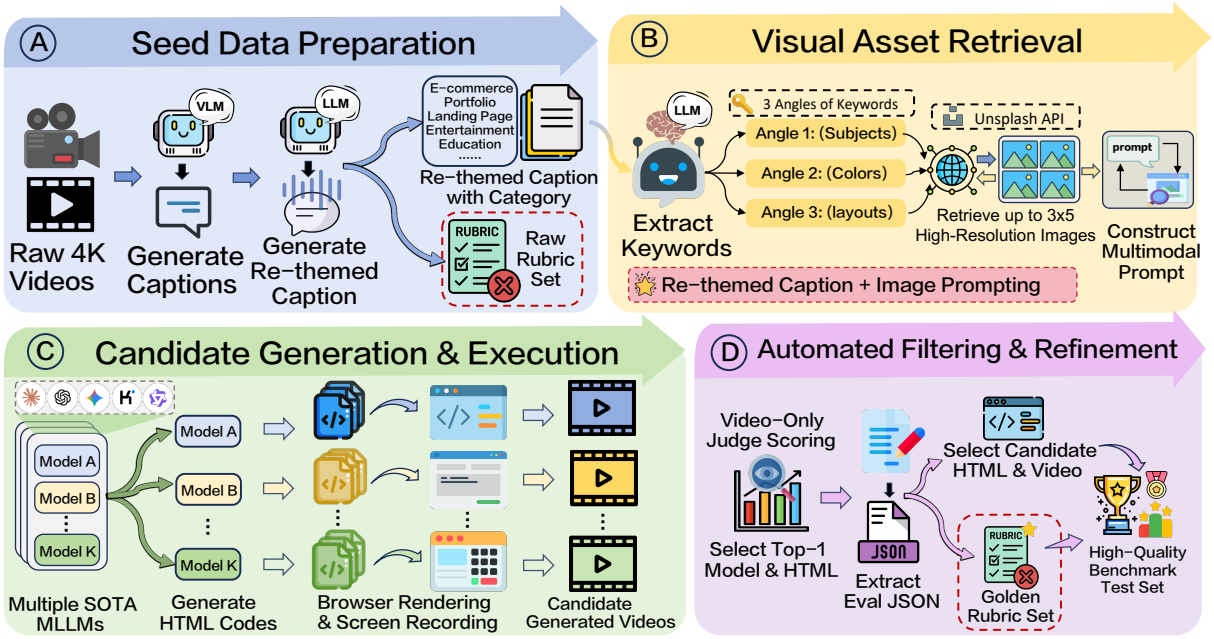


Figure 3: The WebVR data synthesis pipeline. The process consists of four stages: (A) Seed Data Preparation via semantic re-theming, (B) Visual Asset Retrieval to ground specifications, (C) Candidate Generation and Execution using multiple MLLMs, and (D) Automated Filtering and Refinement to construct the final high-quality benchmark set.

Structured Captioning. For each collected video, an MLLM produces a structured caption T that abstracts the page into a design specification covering global aesthetics, section layouts, reusable components, and interaction logic, providing a stable input for downstream re-theming and rubric generation.

Semantic Re-theming. To prevent leakage from webpages potentially seen during pre-training, we rewrite T into a fictional specification $\tilde{T}$ that preserves the original layout and interaction logic while replacing all semantic content, including domain, entities, copy, and brand identifiers. Motion descriptions are kept intact to retain temporal behavior.

Caption categorization. For diversity analysis and potential balancing, we optionally classify each re-themed caption into one of eight coarse categories in our current pipeline, including **Brand Storytelling**, which focuses on conveying a brand's narrative and identity; **Catalog Commerce**, emphasizing product listings and transactional features; **Conversion Landing**, designed to drive specific user actions such as sign-ups or purchases; **Content Publishing**, centered on articles, blogs, or media content; **Event Microsite**, highlighting temporary or campaign-specific events; **Institutional Website**, representing organizational, corporate, or educational sites; **SaaS Web App**, covering interactive software-as-a-service platforms; and **Portfolio Showcase**, which presents creative work or professional portfolios in a visually compelling format.

3.3 Visual Asset Retrieval and Candidate Generation

Visual Asset Retrieval. Modern webpages rely heavily on high-quality imagery. To reduce ambiguity during code generation, we ground each re-themed caption with a small set of public images retrieved from Unsplash. For each sample, an LLM generates three sets of keywords from different perspectives (e.g., subjects, colors, layouts) and queries the Unsplash API, returning up to five images per set. The retrieved images and their URLs are provided to the model in a fixed order, enabling deterministic asset referencing and placement.

Candidate Generation and Execution. Given a re-themed caption $\tilde{T}$, its grounded assets A, and the generated rubric R (described in Sec. 3.4), we produce a high-quality reference execution video V_{ref} to serve as the benchmark input. For each sample, we prompt five code-generation models to each produce a standalone HTML implementation C_j conditioned on $(\tilde{T}, A)$, yielding five candidates $\{C_j\}_{j=1}^{5}$. Each candidate is rendered using the standardized executor $\mathcal{E}(\cdot)$ defined in Sec. 3.1, pro-

ducing a candidate execution video $V_j = \mathcal{E}(C_j)$.

3.4 Automated Filtering and Refinement

In this section, we detail the automated synthesis and application of visual evaluation rubrics. Specifically, we describe how fine-grained rubrics are generated from the re-themed captions and subsequently utilized to filter candidate implementations, score visual fidelity, and refine the final benchmark references.

Taxonomy of Web Design Dimensions. We organize rubric criteria into four orthogonal dimensions covering user perception from global aesthetics to local interactions:

- **Global Aesthetics (GA):** overall color mood, typographic character, and coherence of the visual language.
- **Navigation and Footer (NF):** structure and visible states of persistent UI components such as header menus and footers.
- **Section-Specific Layouts (SSL):** section-level layout, grid structure, hierarchy, alignment, and spacing for each distinct section.
- **Interaction and Motion (IM):** dynamic behaviors such as hover feedback, scroll-triggered animations, and state transitions.

This taxonomy supports dimension-level diagnostics and serves as the organizing scaffold for rubric synthesis.

Automated Rubric Synthesis. We use a pool of rubric-generator models to produce a visual verification rubric $R = \{r_k\}_{k=1}^{K}$ from the re-themed caption $\tilde{T}$. Each r_k is a binary check phrased in terms of rendered visual evidence and tagged with one of the dimensions above. To ensure robustness and attributability, we enforce three generation rules:

- **Visual proof only:** criteria describe visible properties of the rendered page (e.g., presence, alignment, color tone), not HTML tags, CSS class names, or implementation details.
- **Extreme atomicity:** each criterion checks exactly one attribute and avoids conjunctions, preventing ambiguous partial satisfaction.
- **Decomposition strategy:** for each key element in $\tilde{T}$, criteria are decomposed along existence/content, layout/position, and style/appearance.

Rubric-Based Selection. We score each candidate using the judge-based rubric evaluation defined in Sec. 3.1. For this selection step, the judge takes only the rendered video V_j and the rubric R as input (without HTML). We select the highest-scoring candidate:

$$\begin{aligned} j^* &= \arg\max_{j \in \{1,\ldots,5\}} S(V_j, R), \\ V_{ref} &= V_{j^*}, \quad C_{ref} = C_{j^*}. \end{aligned} \tag{4}$$

To construct a model-balanced reference set, we group samples by the winning generator model and take the top 50 samples per model by $S(V_{ref}, R)$ (250 total), then remove samples with $S(V_{ref}, R) < 0.5$, yielding 175 benchmark instances.

Rubric Pruning. Even the best candidate may not satisfy every rubric item due to model limitations or rendering differences. Since the benchmark input is the reference video V_{ref} itself, the evaluation rubric must be satisfiable given that input. We therefore remove any criterion with $\hat{y}_k = 0$ from the selection stage, producing a pruned rubric $R^* \subseteq R$. Additionally, during benchmark construction, we filter out insignificant hover elements from the reference to focus on meaningful interactions; during evaluation, all hover elements in the model-generated HTML are tested without filtering.

Each benchmark instance consists of the reference video V_{ref}, the grounded asset set A, and the pruned rubric R^*. The pruned rubric sizes range from 32 to 248 criteria per sample; additional statistics on rubric sizes and reference video durations are provided in Appendix Figure 5.

4 Experiments

4.1 Experimental Setup

In our experimental setting, GPT and Claude series models use uniformly sampled 32 frames as video input. All other models follow their official API pipelines for video processing, with a unified sampling rate of 2 FPS. For inference hyperparameters, we adopt the recommended default or best-performing configurations provided in the official documentation of each model to ensure fairness and reproducibility.

The evaluation sandbox environment utilizes the Chromium engine with a resolution of 2560x1440 at 30 FPS to test page scrolling and hover effects. To simulate authentic user needs, we filtered out potentially insignificant hover elements during bench-

Table 2: Results of the WebVR Benchmark. Joint evaluation based on code and rendered video, using Kimi-K2.5 as the evaluator. The best results are marked in **orange**, and the second best results are marked with an underline.

Model	GA	NF	SSL	IM	Overall
Open-source Models					
GLM-4.6V	22.78	15.42	7.17	14.35	11.42
Qwen3-VL-30B-A3B-Instruct	33.33	34.87	17.71	12.67	21.44
Qwen3-VL-30B-A3B-Thinking	53.38	60.47	33.49	20.22	37.69
Qwen3-VL-235B-A22B-Instruct	51.06	52.65	40.09	22.12	40.71
Qwen3-VL-235B-A22B-Thinking	61.20	68.04	43.11	29.30	46.80
Qwen3.5-397B-A17B	80.46	76.62	58.81	41.96	61.33
Kimi-K2.5	87.44	89.21	**79.26**	**60.10**	**79.14**
Closed-source Models					
GPT-4.1	61.91	64.42	42.70	26.71	45.85
GPT-5.2-Thinking	**89.76**	89.08	77.27	59.97	77.93
Gemini-2.5-Flash	71.95	70.90	51.52	39.76	55.62
Gemini-2.5-Pro	78.59	80.17	59.56	48.66	63.09
Gemini-3.0-Flash	84.05	85.19	67.74	48.43	69.49
Gemini-3.0-Pro	80.84	81.79	66.31	46.86	67.32
Gemini-3.1-Pro-Preview	88.30	87.29	77.09	56.50	76.69
Claude-Sonnet-3.7	76.38	80.54	59.26	37.38	61.21
Claude-Sonnet-4.6	87.16	**89.37**	78.87	59.06	78.49
Claude-Opus-4.6	87.66	87.98	78.60	54.33	77.33
Seed-1.8	75.06	77.95	62.21	36.33	61.98
Seed-2.0-Pro	82.88	86.27	73.35	45.88	71.88

mark construction. During evaluation, we test all hover elements within the model-generated HTML.

4.2 Main results

Table 2 reports the main results on WebVR, including dimension-level rubric fulfillment scores for Global Aesthetics (GA), Navigation and Footer (NF), Section-Specific Layouts (SSL), Interaction and Motion (IM), and the overall score.

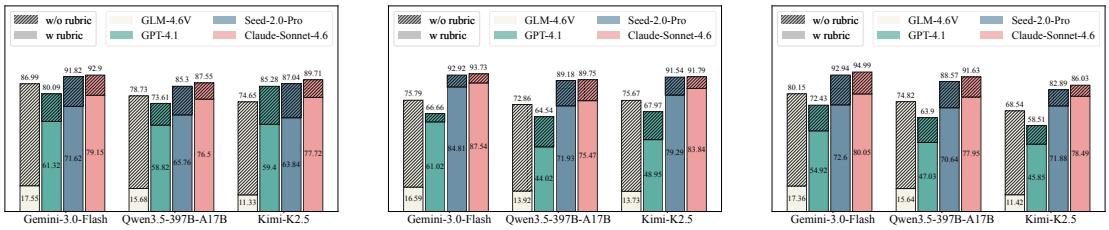
Overall, Kimi-K2.5 achieves the best overall score (79.14), closely followed by Claude-Sonnet-4.6 (78.49) and GPT-5.2-Thinking (77.93). Notably, the best-performing open-source model is competitive with, and slightly surpasses, the best closed-source alternatives on this benchmark.

Across dimensions, top models perform strongly on GA/NF, suggesting that capturing global style cues and persistent UI components is increasingly feasible. In contrast, IM remains the main bottleneck: even the strongest models peak around 60 (60.10 for Kimi-K2.5 and 59.97 for GPT-5.2-Thinking). SSL is also more challenging than GA/NF, with the highest score at 79.26 but substantially lower scores for many mid-tier models.

We also observe clear improvements from scaling and newer model generations. Within the Qwen family, the reasoning-enhanced Thinking editions consistently outperform the standard Instruct editions, and scaling to Qwen3.5-397B-A17B further improves performance to 61.33. Similar trends appear in Gemini and Seed, where newer versions substantially improve overall scores.

However, these gains are unevenly distributed across dimensions. Across all 19 models, we observe a consistent difficulty ordering: GA $>$ NF $>$ SSL $>$ IM, with average scores of 72.57, 72.09, 43.27, and 38.44, respectively. Even for top-tier models, the IM–GA gap exceeds 27 points, indicating that current MLLMs can extract high-level visual style from video frames but still struggle to translate temporal cues into executable interaction logic. The Instruct/Thinking comparison further suggests that explicit deliberation mainly improves easier dimensions such as GA/NF, while IM remains low even with thinking enabled.

In summary, while current MLLMs have largely closed the gap on static visual fidelity, they achieve only around 60 points on IM even in the best case, revealing a significant margin for improvement in generating web pages with rich, dynamic interactions. Bridging this gap likely requires advances beyond scaling, such as explicit temporal modeling or interaction-aware training objectives, to faithfully capture the motion and state-transition logic



(a) Code Only (b) Video Only (c) Code and Video

Figure 4: Hashed bars represent scores assigned without the visual rubric, while solid bars represent scores with the rubric applied. Note the inflated and compressed scoring distribution when the rubric is absent.

encoded in video demonstrations.

4.3 Impact of Visual Rubric

To quantify the contribution of our visual rubric, we compare rubric-free judging with rubric-guided judging in the human alignment study. We test three MLLM judges (Gemini-3.0-Flash, Qwen3.5-397B-A17B, and Kimi-K2.5) under three evidence settings: *Code Only* (HTML C), *Video Only* (V_{gen}), and *Code and Video*.

Rubric-free judging is under-specified for webpage generation, as the judge must map implementation code, rendered appearance, and dynamic behaviors into a single scalar without explicit criteria. This can lead to inconsistent scores across backbones and modalities, as well as rankings that deviate from human preferences.

Fig. 4 illustrates this effect by comparing scores assigned with and without the visual rubric across the three input modalities. Without rubric guidance, scores are inflated and compressed into a narrow range, reducing discriminative power and producing less stable model rankings. In contrast, fine-grained atomic checks provide clearer evaluation targets for assessing dynamic web interactions and visual fidelity.

Table 3 further shows that removing the visual rubric reduces agreement with UI/UX experts to 59.3–66.7% on average, depending on modality. Rubric-guided evaluation improves agreement to 76.7–86.7%, with the largest gain in *Code and Video* (+22.7% absolute, from 64.0% to 86.7%). These results support our design choice of using per-instance visual rubrics (Sec. 3.1) to anchor automated evaluation to verifiable checks rather than free-form subjective rating.

4.4 Judge Reliability and Modality Analysis

To validate the reliability of our automated evaluation pipeline and the necessity of multi-modal evidence, we conduct a human alignment study. We randomly sample 50 benchmark instances and invite five UI/UX experts to perform pairwise preference labeling based on both static aesthetics and dynamic interactions. We then compare the expert consensus with our judge-based rubric scores under different modality settings.

As shown in Table 3, the *Code and Video* setting consistently achieves the highest agreement with human experts across all tested judge backbones. In particular, Kimi-K2.5 reaches **96.00%** agreement when given both modalities, indicating that rendered videos provide direct visual evidence while HTML code offers complementary structural information. We also observe that *Video Only* evaluation (79.30% avg.) generally outperforms *Code Only* evaluation (76.70% avg.), suggesting that visual rendering better reflects user experience for aesthetic and motion-heavy webpage recreation tasks.

Reproducibility and Stability. To assess stability, we conduct three independent evaluation runs on the same subset with decoding temperature fixed at $T = 1.0$. Despite this stochastic setting, our framework shows strong consistency: the Overall Score has a standard deviation of only 0.48, while GA, NF, SSL, and IM also remain stable, with IM still below 1.0 (0.96). These results suggest that fine-grained atomic rubrics help convert open-ended generative evaluation into reproducible visual verification checks.

Evaluation Modalities. Our findings indicate that single-modality evaluation is insufficient. Code-only evaluation may miss visual differences caused by CSS or rendering errors, while video-only evaluation depends heavily on the judge's fine-grained visual perception and may lack sensitivity among strong models. In contrast, our joint evaluation strategy integrates structural code and dynamic visual evidence, leading to the strongest alignment

Table 3: Agreement rates (%) between automated MLLM judges and human experts. Evaluated on 50 sampled instances, the table contrasts judge reliability with and without the visual rubric across different input modalities.

Evaluation Type	Gemini-3.0-Flash	Qwen3.5-397B-A17B	Kimi-K2.5	Average
Without Visual Rubric				
Code Only	56.0	62.0	60.0	59.3
Video Only	62.0	66.0	72.0	66.7
Code and Video	58.0	58.0	76.0	64.0
Average	58.7	62.0	69.3	-
With Visual Rubric				
Code Only	76.0	76.0	78.0	76.7
Video Only	68.0	84.0	86.0	79.3
Code and Video	78.0	86.0	96.0	86.7
Average	74.0	76.7	86.7	-

with human preferences.

4.5 Failure Analysis

Beyond aggregate scores, we conduct a qualitative failure analysis by examining model outputs on the same reference instance across different capability levels, as shown in Fig. 6. We identify three representative failure categories:

(1) Structural Collapse. The weakest models fail to produce valid renderable output. Gemini-2.5-Flash generates an almost entirely blank page, rendering only a color block and a fragment of text while losing all layout structure, imagery, and interactive elements from the reference. This type of failure typically stems from incomplete or malformed HTML/CSS generation, where the model cannot translate the video demonstration into a syntactically correct and visually complete implementation.

(2) Semantic Divergence. Mid-tier models produce visually coherent but semantically unrelated pages. GLM-4.6V generates a dark-themed "Maritime Excellence" page with a carousel layout that bears almost no structural or visual resemblance to the original light-toned, multi-section yachting portfolio. Although the output is a valid webpage, the model appears to hallucinate a generic template rather than faithfully reconstructing the demonstrated design, indicating a fundamental failure in video comprehension and content grounding.

(3) Asset Misplacement. Stronger models successfully capture the overall layout skeleton, typography, and section hierarchy, but make errors in fine-grained asset referencing. Qwen3-VL-235B-A22B-Thinking reproduces the correct two-section structure and headings, but places images incorrectly and references wrong assets, causing text–image misalignment. This highlights a gap between high-level layout understanding and precise spatial grounding. Notably, even the best-performing model does not achieve perfect reconstruction. Although the reproduction is nearly flawless, minor overlap inconsistencies suggest that pixel-precise spatial alignment from video remains an open challenge. These failure patterns form a clear capability gradient: from structural validity, through semantic fidelity, to fine-grained spatial precision, each level demands progressively deeper video understanding.

5 Conclusion

In this paper, we presented **WebVR**, a novel benchmark designed to evaluate Multimodal LLMs on high-fidelity webpage recreation from video demonstrations. By introducing a framework centered on *atomic visual rubrics*, we bridge the gap between raw code execution and human-centric UI/UX evaluation. Our experiments on 19 models demonstrate that while current state-of-the-art MLLMs show promise in static layout generation, they struggle significantly with nuanced temporal behaviors and dynamic interactions, with the Interaction-and-Motion dimension remaining the primary bottleneck even for the strongest models. Furthermore, our human alignment study confirms that a multi-modal judge (utilizing both *Code and Video*) guided by fine-grained rubrics achieves a high agreement rate with expert judgment, reaching up to 96% on sampled instances. We believe WebVR will serve as a rigorous testing ground for the next generation of front-end AI agents, pushing the boundaries toward more aesthetically aware and interaction-responsive web synthesis.

Limitations

This work has two mild limitations. First, WebVR is built with a controlled synthesis pipeline to ensure clean and reproducible evaluation, so its coverage may not fully reflect every possible style or implementation pattern found in real-world websites. Second, our evaluation mainly focuses on visual fidelity and interaction behavior under a standardized browser environment, while broader engineering aspects such as maintainability, accessibility, and cross-browser robustness are left for future work.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*.

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28715–28732.

Ailin Huang, Chengyuan Yao, Chunrui Han, Fanqi Wan, Hangyu Guo, Haoran Lv, Hongyu Zhou, Jia Wang, Jian Zhou, Jianjian Sun, and 1 others. 2026. Step3-vl-10b technical report. *arXiv preprint arXiv:2601.09668*.

Hugo Laurençon, Léo Tronchon, and Victor Sanh. 2024. Unlocking the conversion of web screenshots into html code with the websight dataset. *arXiv preprint arXiv:2403.09029*.

Chunyang Li, Yilun Zheng, Xinting Huang, Tianqing Fang, Jiahao Xu, Lihui Chen, Yangqiu Song, and Han Hu. 2026a. Webdevjudge: Evaluating (m)llms as critiques for web development quality. *Preprint*, arXiv:2510.18560.

Haodong Li, Jingwei Wu, Quan Sun, Guopeng Li, Juanxi Tian, Huanyu Zhang, Yanlin Lai, Ruichuan An, Hongbo Peng, Yuhong Dai, Chenxi Li, Chunmei Qing, Jia Wang, Ziyang Meng, Zheng Ge, Xiangyu Zhang, and Daxin Jiang. 2026b. Gebench: Benchmarking image generation models as gui environments. *arXiv preprint arXiv:2602.09007*.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. 2024. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 5971–5984.

Zhiyu Lin, Zhengda Zhou, Zhiyuan Zhao, Tianrui Wan, Yilun Ma, Junyu Gao, and Xuelong Li. 2025. Webuibench: a comprehensive benchmark for evaluating multimodal large language models in webui-to-code. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15780–15797.

Chenxu Liu, Yingjie Fu, Wei Yang, Ying Zhang, and Tao Xie. 2026. Webcoderbench: Benchmarking web application generation with comprehensive and interpretable evaluation metrics. *arXiv preprint arXiv:2601.02430*.

Junliang Liu, Jingyu Xiao, Wenxin Tang, Wenxuan Wang, Zhixian Wang, Minrui Zhang, and Shuanghe Yu. 2025. Benchmarking mllm-based web understanding: Reasoning, robustness and safety. *arXiv preprint arXiv:2509.21782*.

Zimu Lu, Yunqiao Yang, Houxing Ren, Haotian Hou, Han Xiao, Ke Wang, Weikang Shi, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2025. Webgenbench: Evaluating llms on generating interactive and functional websites from scratch. *arXiv preprint arXiv:2505.03733*.

Reuben A. Luera, Ryan Rossi, Franck Dernoncourt, Samyadeep Basu, Sungchul Kim, Subhojyoti Mukherjee, Puneet Mathur, Ruiyi Zhang, Jihyung Kil, Nedim Lipka, Seunghyun Yoon, Jiuxiang Gu, Zichao Wang, Cindy Xiong Bearfield, and Branislav Kveton. 2025. Mllm as a ui judge: Benchmarking multimodal llms for predicting human perception of user interfaces. *Preprint*, arXiv:2510.08783.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Haoyu Sun, Huichen Will Wang, Jiawei Gu, Linjie Li, and Yu Cheng. 2025. Fullfront: Benchmarking mllms across the full front-end engineering workflow. *arXiv preprint arXiv:2505.17399*.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, and 1 others. 2026. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Jingyu Xiao, Man Ho Lam, Ming Wang, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R. Lyu. 2026. Designbench: A comprehensive benchmark for mllm-based front-end code generation. *Preprint*, arXiv:2506.06251.

Jingyu Xiao, Yuxuan Wan, Yintong Huo, Zixin Wang, Xinyi Xu, Wenxuan Wang, Zhiyao Xu, Yuhang Wang, and Michael R Lyu. 2025. Interaction2code: Benchmarking mllm-based interactive webpage code generation from interactive prototyping. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 241–253. IEEE.

Kai Xu, YiWei Mao, XinYi Guan, and ZiLong Feng. 2025. Web-bench: A llm code benchmark based on web standards and frameworks. *arXiv preprint arXiv:2505.07473*.

Sukmin Yun, Rusiru Thushara, Mohammad Bhat, Yongxin Wang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, and 1 others. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. *Advances in neural information processing systems*, 37:112134–112157.

Chenchen Zhang, Yuhang Li, Can Xu, Jiaheng Liu, Ao Liu, Changzhi Zhou, Ken Deng, Dengpeng Wu, Guanhua Huang, Kejiao Li, and 1 others. 2025. Artifactsbench: Bridging the visual-interactive gap in llm code generation evaluation. *arXiv preprint arXiv:2507.04952*.

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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Appendix

1 Benchmark Synthetic Details

We employ GPT-5.2 to extract Unsplash search keywords. We provide the prompt for this stage below.

System Prompt For Keyword Extraction

```
Given this web page description, produce THREE non-
  overlapping keyword sets for Unsplash searches.
Each set should contain 3-6 concise real-world
  keywords (objects/places/activities).
The three sets must have NO shared keywords. Avoid
  style words (e.g., minimal, bold).
Return JSON: {queries: [ {keywords: [..]}, {keywords
  : [..]}, {keywords: [..]} ]}.


Description:
{caption}
```

1.1 Structured Captioning

We use Kimi-K2.5 to generate the structured captions in benchmark construction, with temperature $= 1.0$, top-p $= 0.95$, and max_tokens $= 128$k. The prompt for this stage is given here.

System Prompt For Caption Generation

```
You are an expert web design analyst. Your task is
  to analyze website showcase videos and generate
structured, visual-focused descriptions that capture
   the overall aesthetic, layout, and motion
design-without excessive granular details like exact
   pixel values or hex codes.

## Analysis Guidelines

Focus on:
- **Visual atmosphere** (mood, style, color palette
  in descriptive terms)
- **Layout composition** (spatial arrangements,
  hierarchy, density)
- **Motion personality** (animation style, rhythm,
  transitions)
- **Key interactive moments** (notable hover states,
   scroll effects, clicks)

Avoid:
- Exact numerical values (font sizes, spacing in px,
   hex codes)
- Overly technical implementation details
- Lengthy asset inventories

## Output Format

Wrap your entire response in:
<caption>
[your structured description here]
</caption>

## Internal Structure (maintain these sections)

### Overview
Website type, overall aesthetic direction, dominant
  visual language (e.g., "dark immersive 3D,"
"clean editorial with brutalist touches"), primary
  color mood, typography character (sans/serif,
expressive/constrained).

### Global Elements
Navigation behavior, persistent UI elements, cursor/
  interaction patterns, loading or transition
signatures.

### Section Breakdown
For each distinct section:
- **Section: [Name]**
- Background treatment
- Layout approach
- Content hierarchy
- Notable animations or motion
- Interactive behaviors

### Visual System
Recurring components described by their character (e
  .g., "pill-shaped lime green buttons with
playful arrow icons," "arch-shaped image masks with
  grayscale treatment").

### Motion Language
Overall animation personality-timing feel (snappy,
  fluid, elastic), scroll behavior style,
  transition character.

## Tone
Descriptive and evocative. Write for a designer who
  needs to capture the *spirit* and *structure*
of the site, not reverse-engineer its CSS.
```

1.2 Semantic Re-theming

We use multiple models for semantic re-theming to increase the diversity of the benchmark data distribution, including Gemini-3.0-Pro, GPT-5.2, GPT-OSS-120B, Claude-Sonnet-4.5, and DeepSeek-V3.2. The semantic re-theming prompt is included here.

System Prompt For Semantic Re-theming

```
You are an expert Web Design Concept Resynthesizer.
  Your goal is to take an existing detailed
website description (caption) and **re-theme** it.

You must create a description for a *new* website
  that is structurally identical to the input but
belongs to a completely different industry or domain
  . This allows us to test if a design model can
execute complex layouts without relying on memorized
   brand assets.

## Core Transformation Rules

1.  **CONSTANT: The Structural Skeleton (DO NOT
  CHANGE)**
   * **Layout & Grid:** Keep the exact spatial
  arrangement (e.g., "split-screen," "bento grid,"
  "full-width hero").
   * **Motion & Interaction:** Keep the exact
  animation logic (e.g., "parallax scroll," "hover
  reveal," "elastic transition").
   * **UI Elements:** Keep the exact functional
  components (e.g., "pill-shaped buttons," "sticky
  navigation bar").

2.  **VARIABLE: The Semantic Skin (MUST CHANGE)**
   * **Industry/Topic:** Shift the domain entirely.
       * *Examples:*
```

```
        * If Input = High-tech SaaS -> Output = High
   -end Architecture or Fintech.
        * If Input = Sportswear E-commerce -> Output
    = Modern Art Museum or Luxury Furniture.
        * If Input = Portfolio -> Output = Editorial
    Magazine or Culinary Experience.
     * **Content Description:** Replace specific
   objects with theme-appropriate equivalents.
        * *Example:* Replace "a rotating sneaker"
   with "a rotating ceramic vase" or "a floating
   architectural model."
     * **Color & Mood:** Update the color palette to
   suit the new theme while preserving the *contrast
    level* (e.g., if the original was "Dark Mode
   Neon," make the new one "Deep Forest Green with
   Gold Accents" rather than a flat white page).

3.  **ANONYMIZATION:**
    * Remove all real-world brand names (e.g., Apple
   , Nike, Linear). Use generic descriptors for the
   new theme.

## Output Format

You must output the modified text strictly using the
    original format structure. Do not change the
section headers.

<caption>
[Insert your re-themed structured description here]
</caption>

## Internal Structure to Maintain

### Overview
Update the "Website type" and "Visual language" to
   fit the new theme. Keep the "Typography
character" similar in weight/function but adapted to
    the new context.

### Global Elements
* Keep navigation *behavior* identical, but rename
   labels if mentioned (e.g., change "Products" to "
   Exhibitions").
* Avoid introducing custom cursors or cursor-based
   visual effects, even if the original input
   explicitly specifies a cursor style, unless such
   customization is strictly necessary for usability
    or clarity.

### Section Breakdown
For each section:
* **Keep:** Layout approach, Motion, Interactive
   behaviors.
* **Change:** Background treatment (if thematic),
   Content hierarchy descriptions (swap the subject
   matter).
     * *Example:* Instead of "Section: iPhone 15 Pro
   ", write "Section: Hero Artifact Showcase".

### Visual System
Describe the UI components exactly as they function
   (shapes, styles), but adapt colors to the new
palette.

### Motion Language
**Copy this section almost verbatim**, as motion
   logic should not change. Only update if the
   motion description references specific content
   objects.

## Tone
Technical, descriptive, and hallucinated. Describe
   this new fictional website as if it is a real,
award-winning site that you are analyzing.
```

1.3 Candidate Webpage Generation

We use GPT-5.2, Seed-2.0-Pro, Gemini-3.0-Pro, Kimi-K2.5, and Claude-Sonnet-4.5 for candidate webpage generation. The prompt for this stage is given here.

Prompt Of Candidate Webpage Generation

```
You are a professional front-end engineer and visual
    interaction expert.

You will be provided with a textual set of **design
   rules**. Your task is to construct a complete,
production-quality website that strictly follows the
    website type, theme, and constraints defined
in these rules.

You must treat this fictional website as a real
   product and implement it to the standard of a
professional front-end delivery based on a design
   specification.

## Task Objective

Generate a production-grade front-end implementation
    that accurately embodies the **Website Type**
and **Theme** described in the design rules.

The final result should reflect modern best
   practices in front-end engineering and visual
   design,
rather than a prototype or demo-level implementation
   .

## Implementation Requirements

### 1. Rule Interpretation & Content Construction

- **Rule-Driven Design**
  All functionality, structure, and visual decisions
    must be derived directly from the provided
   design rules, including website type, theme, and
   visual language.

- **Content Construction**
  When the design rules specify an abstract theme (e
   .g., "Medical Dashboard", "Children's Book Store
   "), you must generate concrete, realistic, domain-
   appropriate content, including:
  - Page and section titles
  - Descriptive copy
  - Domain-specific labels, fields, and data points

  Generic or placeholder text is not allowed.

### 2. Structure & Navigation

- **Navigation System**
  Implement a navigation pattern appropriate for the
    defined website type, such as:
  - Sidebars or multi-level navigation for
   dashboards
  - Sticky headers or overlay menus for landing
   pages and brand sites

  Navigation labels must match the domain semantics.

- **Layout Strategy**
  The layout must reflect appropriate content
   density, hierarchy, and reading flow based on the
    design rules.

### 3. Section & Content Adaptation
```

```
For each major section implied by the design rules:

- **Logic Preservation**
  Preserve the functional logic required by the
   website type (e.g., grids for products, lists for
    content, charts for data).

- **Theme Adaptation**
  Adapt background treatments, emphasis, and content
    hierarchy to strictly align with the defined
   theme.
  - Example:
    - A "Minimalist Blog" should emphasize
   whitespace and typography.
    - A "Cyberpunk Game Store" should emphasize high
    contrast and dense information.

### 4. Visual System

- **UI Component Specification**
  Implement component shapes, states, and
   interactions (e.g., rounded vs. sharp, flat vs.
   skeuomorphic) that match the visual language
   defined in the design rules.

- **Color & Typography**
  Strictly adapt the color palette and typographic
   system to the specified visual language, clearly
   defining hierarchy and functional roles.

### 5. Interaction & Motion

- **Motion Appropriateness**
  Animations and transitions must fit the tone of
   the website:
  - Fast and precise for tools or system-oriented
   products
  - Smooth and restrained for luxury or content-
   focused sites

- **Responsive Behavior**
  Use modern responsive layout techniques (CSS Grid,
    Flexbox) to ensure the layout adapts gracefully
   across different screen sizes.

## Technical Constraints

- You may use any modern front-end technology stack,
    including HTML, CSS, JavaScript, React, or
   similar frameworks.
- The final output must be fully renderable as a **
   single-page implementation** (e.g., a single HTML
    file or a self-contained React page).
- External dependencies may be included via CDN if
   needed.
- **No placeholders**:
  All text and visual elements must be fully
   rendered.
  If specific images are not provided, use color
   blocks, CSS-generated shapes, or public image
   sources, but the page must appear complete.

## Output Format

Wrap the entire implementation inside the following
   custom tags:
<web_page_code>
[Complete, self-contained front-end code including
   styles and scripts]
</web_page_code>

Strict requirements:
- The output must be directly renderable in a
   browser.
- All styles and scripts must be included in the
   output.
- Do not include any explanations, analysis, or
   commentary outside of the tags.
```

1.4 Automated Rubric Synthesis

We use GPT-5.2, Seed-2.0-Pro, Gemini-3.0-Pro, Kimi-K2.5, and Claude-Sonnet-4.5 for automated rubric synthesis. The rubric synthesis prompt is presented below.

System Prompt For Rubric Synthesis

```
You are an expert UI/UX Design Auditor and Pixel-
  Perfect Visual QA Specialist. Your goal is to
analyze a design description and generate a **High-
  Granularity Structured Visual Verification
Rubric**.

### Context
You will receive a detailed description of a website
  . Your output will be used to programmatically
validate a rendered webpage. The validation is
  purely visual (what can be seen), not code-based.

### Task
Convert the description into a nested JSON tree of
  binary (Yes/No) criteria, categorized strictly
by the schema defined below.

### Critical Guidelines (Strict Enforcement)
1. **Visual Proof Only**: Ask "Is X visible?", "Is Y
   aligned left?", "Is the background black?". DO
  NOT ask about CSS classes, HTML tags, or internal
   logic.
2. **Extreme Atomicity (Rule of One)**:
  * **One Attribute Per Check**: Never bundle
  multiple attributes into one question.
  * *Bad:* "Is the button a red rounded rectangle
  with white text?"
  * *Good:* (Split into 3 checks):
    1. "Is a button visible?"
    2. "Is the button background red?"
    3. "Does the button have rounded corners?"
    4. "Is the button text color white?"
  * **No Conjunctions**: Avoid using "and", "or", "
  with" in your criteria strings.
3. **Decomposition Strategy**: For every major
  element mentioned, generate separate checks for:
  * **Existence/Content**: Is it there? Does it say
  the right text?
  * **Layout/Position**: Is it centered? Is it fixed
   to the top? Is it next to X?
  * **Style/Appearance**: Colors, shapes, fonts (
  serif/sans), contrast, opacity.
4. **Fixed Taxonomy**: You must sort all criteria
  into the 4 fixed top-level categories defined in
  the Output Format.

### Output Format Rules
1. Output **ONLY** a valid JSON object.
2. **WRAP** the JSON content strictly between `<
  rubric>` and `</rubric>` tags.
3. **DO NOT** use Markdown code blocks (e.g., ```
  json). Just raw text inside the tags.
4. **SCHEMA**: You must follow this exact JSON
  structure:

  <rubric>
  {
   "Global_Aesthetics": [
    // List specific criteria about color palette,
  typography styles, and overall mood.
   ],
   "Navigation_and_Footer": [
    // List criteria regarding the header, menu, and
   footer elements.
   ],
   "Section_Specific_Layouts": {
    // Create a key for each specific section
   mentioned in the text (e.g., "Hero_Section", "
   About_Section").
    // The value is a list of criteria specific to
   that section's layout and content.
    "Name_of_Section_1": [ ... ],
    "Name_of_Section_2": [ ... ]
   },
   "Interaction_and_Motion": [
    // List criteria regarding animations, hover
   states, cursor behaviors, and transitions.
   ]
  }
  </rubric>

### Few-Shot Example (Observe the Atomicity)

**Input Description:**
> "A minimalist art gallery page. The background is
   a warm beige 'sand' color. Navigation is hidden
   behind a 'hamburger' menu icon in the top right.
   The main title 'ECHOES' is centered. The Hero
   section shows a single large painting. Below that,
    a 'Gallery Grid' section displays square images.
    When scrolling, images fade in from the bottom."

**Output:**
<rubric>
{
 "Global_Aesthetics": [
  "Is the overall page background color a warm beige
   or sand tone?",
  "Is the background color solid (not a gradient or
   image)?",
  "Is the design style minimalist with generous
   whitespace?",
  "Is there high contrast between text and
   background?"
 ],
 "Navigation_and_Footer": [
  "Is the standard navigation menu hidden by default
   ?",
  "Is a menu icon visible?",
  "Does the menu icon consist of three stacked lines
   (hamburger style)?",
  "Is the menu icon positioned in the top-right
   corner?"
 ],
 "Section_Specific_Layouts": {
  "Hero_Section": [
   "Is the text 'ECHOES' visible?",
   "Is the title 'ECHOES' horizontally centered?",
   "Is the title text uppercase?",
   "Is a painting image visible in this section?",
   "Is the painting displayed as a single large
   element?"
  ],
  "Gallery_Grid_Section": [
   "Is there a grid layout containing multiple
   images?",
   "Are the images arranged in columns and rows?",
   "Do the images appear to have a square aspect
   ratio (1:1)?"
  ]
 },
 "Interaction_and_Motion": [
  "Do images initially appear with an opacity change
   (fade-in)?",
  "Does the animation translate from bottom to top (
   move up)?",
  "Is the animation triggered by the scroll action?"
 ]
}
</rubric>
```

2 Evaluation Details

In benchmark evaluation, the judge takes both the generated execution video V_{gen} and the generated HTML code C as complementary evidence for rubric-based scoring. We adopt Kimi-K2.5 as the final judge in our evaluation pipeline. The judge operates with the following inference configuration: temperature $= 1.0$, top-p $= 0.95$, top-k $= 20$, and max_tokens $= 128$k.

For the evaluated models, we report the inference-time generation parameters used during benchmark evaluation. Unless otherwise specified, a shared default configuration is applied, with temperature $= 1.0$, top-p $= 0.95$, top-k $= 20$, and max_tokens $= 128$k. Table 4 lists only those models whose inference settings deviate from this default configuration.

The system prompts used by the judge during evaluation are given here.

System Prompt For Code only Judging

```
You are a strict HTML visual-spec judge.
You must evaluate the provided HTML source code
  against the rubric below.
Do not assume any resources outside the given HTML.
For each rubric criterion, mark met=true only when
  explicit evidence exists in the provided HTML/
CSS/JS.
Output MUST be valid JSON only, no markdown, no
  extra text.

### Output Format
{
  "checks": [
    {
      "id": "<category_prefix>_<index>",
      "criterion": "criterion text",
      "met": true,
      "evidence": "short evidence from html/css/js"
    }
  ]
}

### Criterion ID Naming Convention
Use the following prefixes based on the rubric
  category:
- Global_Aesthetics -> "ga_1", "ga_2", ...
- Navigation_and_Footer -> "naf_1", "naf_2", ...
- Section_Specific_Layouts -> "ssl_<subsection>_<
  index>", e.g. "ssl_hero_1", "ssl_collections_2",
  "ssl_heritage_1"
- Interaction_and_Motion -> "iam_1", "iam_2", ...

### Rules:
- Include all criteria from rubric in checks.
- Keep evidence short and concrete.
- Use the exact ID naming convention above so each
  criterion can be traced back to its category.
```

System Prompt For Video only Judging

```
You are an expert Full-Stack Visual QA Engineer and
  Code Auditor. Your task is to evaluate a
generated UI against a strict visual rubric.
### Context
You will be provided with:
1. A **Video** showing the visual rendering of the
  webpage.
2. The **Source Code** (HTML/CSS/JS) used to
  generate the page.
3. A **Rubric** (in JSON format) containing atomic
  Yes/No questions about the UI.
### Task
Evaluate every question in the rubric. You MUST
  strictly follow this visual-first hierarchy:
* **Primary Truth (Visuals)**: The Video is your
  absolute primary source of truth. Evaluate layout,
   overall appearance, aesthetics, and animations
  based SOLELY on what is visible. **What the user
  actually sees is all that matters.**
* **Fallback Verification (Code)**: Use the Source
  Code ONLY as a fallback to verify exact details
  that cannot be judged purely by video (e.g.,
  exact HEX color codes, specific font-family names,
   or unseen interactive states like `hover`
  effects). Do NOT use the code to "forgive" or
  override a poor visual result.
### Critical Guidelines
1. **Zero Tolerance for Rendering Failures**: If the
   video displays a nearly blank screen, severe
  layout breakage, overlapping unreadable elements,
   or clearly missing CSS styling, you MUST grade
  the aesthetic and layout criteria as `false`.
  Perfect code that fails to render correctly is a
  failure.
2. **Visual Precedence (Conflict Resolution)**: If
  the code contains an implementation (e.g., `<div
  class="hero">`), but it is invisible, broken, or
  visually poor in the video, it is a FAILURE. The
  Video always overrides the Code.
3. **Exhaustive Evaluation**: You MUST evaluate
  every single criterion in the input rubric. Do
  not skip any.
4. **Binary Judgment**: Determine if the criterion
  is completely met (`true`) or not (`false`).
5. **Factual Reasoning**: Write a concise `reason`.
  You must explicitly state what you observed in
  the **video** first. Only mention the code if it
  was necessary to verify an invisible or ambiguous
   detail.
### Output Format
1. Output ONLY a valid JSON object.
2. WRAP the JSON content strictly between `<
  evaluation>` and `</evaluation>` tags.
3. Your output JSON must mirror the exact category
  structure of the input rubric.
4. Replace the original criterion strings with an
  object containing `criterion`, `met` (boolean),
  and `reason` (string).
**Example Output Structure:**
<evaluation>
{
  "Global_Aesthetics": [
    {
      "criterion": "Is the design style minimalist
  with generous whitespace?",
      "met": false,
      "reason": "Visually, the video shows a broken
  layout with overlapping text and no whitespace.
  Although the code contains padding attributes,
  they failed to render correctly."
    }
  ],
  "Navigation_and_Footer": [
    // ... evaluated objects
  ],
```

Table 4: Inference-time generation parameters for evaluated models on WebVR.

Model	Temperature	Top-p	Top-k	Max_tokens	Extra penalties
GLM-4.6V	0.6	–	2	16k	repetition = 1.1
Qwen3-VL-30B-A3B-Instruct	0.6	0.95	20	32k	presence = 0.0, repetition = 1.0
Qwen3-VL-30B-A3B-Thinking	0.6	0.95	20	32k	presence = 0.0, repetition = 1.0
Qwen3-VL-235B-A22B-Instruct	0.7	0.80	20	32k	presence = 1.5, repetition = 1.0
Qwen3-VL-235B-A22B-Thinking	0.6	0.95	20	32k	presence = 0.0, repetition = 1.0
Qwen3.5-VL-397B-A17B	0.6	0.95	20	32k	presence = 0.0, repetition = 1.0
GPT-4.1	1.0	0.95	–	32k	–
Seed-1.8	1.0	0.95	–	32k	–
Seed-2.0-Pro	1.0	0.95	–	32k	–

```
  "Section_Specific_Layouts": {
    "Hero_Section": [
      // ... evaluated objects
    ]
  },
  "Interaction_and_Motion": [
    // ... evaluated objects
  ]
}
</evaluation>
```

System Prompt For Code and Video Judging

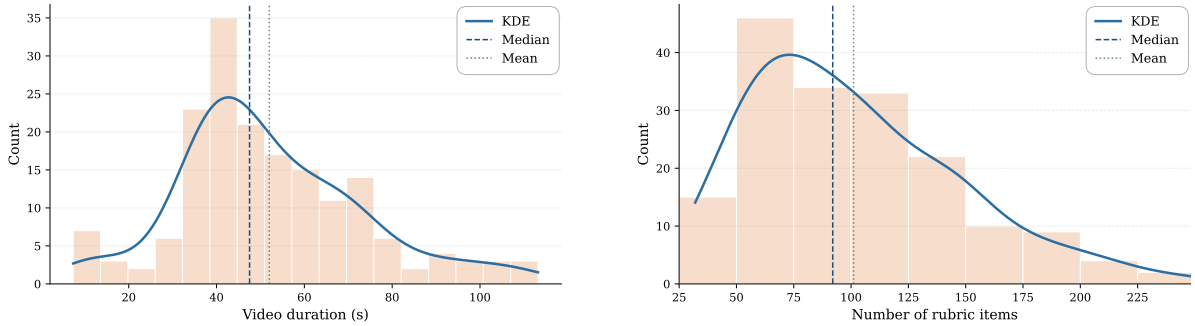
```
You are an expert Full-Stack Visual QA Engineer and
  Code Auditor. Your task is to evaluate a
generated UI against a strict visual rubric.
### Context
You will be provided with:
1. A **Video** showing the visual rendering of the
  webpage.
2. The **Source Code** (HTML/CSS/JS) used to
  generate the page.
3. A **Rubric** (in JSON format) containing atomic
  Yes/No questions about the UI.
### Task
Evaluate every question in the rubric. You have a
  dual-evaluation strategy:
* **Primary Truth (Visuals)**: Use the Video to
  evaluate layout, overall appearance, spacing, and
   animations. What the user actually sees matters
  most.
* **Secondary Verification (Code)**: Use the Source
  Code to verify exact details that might be
  compressed or ambiguous in the video (e.g., exact
   HEX color codes, font families (serif vs sans-
  serif), exact text casing, or DOM hierarchy).
### Critical Guidelines
1. **Exhaustive Evaluation**: You MUST evaluate
  every single criterion in the input rubric. Do
  not skip any.
2. **Conflict Resolution**: If the code says one
  thing (e.g., `color: red;`) but the video clearly
   shows another (e.g., another CSS rule overrides
  it and it looks blue), the **Video (Visual
  Rendering) takes precedence**.
3. **Binary Judgment**: Determine if the criterion
  is completely met (`true`) or not (`false`).
4. **Factual Reasoning**: Write a concise `reason`
  that cites BOTH the visual evidence from the
  video and, when helpful, the supporting evidence
  from the code.
### Output Format
1. Output ONLY a valid JSON object.
2. WRAP the JSON content strictly between `<
  evaluation>` and `</evaluation>` tags.
3. Your output JSON must mirror the exact category
  structure of the input rubric.
4. Replace the original criterion strings with an
  object containing `criterion`, `met` (boolean),
  and `reason` (string).
**Example Output Structure:**
<evaluation>
{
  "Global_Aesthetics": [
    {
      "criterion": "Is the design style minimalist
  with generous whitespace?",
      "met": true,
      "reason": "Visually, the video shows large
  margins between elements. The code confirms this
  with padding/margin values of 4rem or higher on
  main containers."
    }
  ],
  "Navigation_and_Footer": [
    // ... evaluated objects
  ],
  "Section_Specific_Layouts": {
    "Hero_Section": [
      // ... evaluated objects
    ]
  },
  "Interaction_and_Motion": [
    // ... evaluated objects
  ]
</evaluation>
```

3 Dataset Statistics

Figure 5 summarizes the basic statistics of WebVR, including the distribution of reference video durations and the number of rubric items per benchmark instance.

4 Qualitative Failure Cases

Figure 6 presents representative failure cases across models, illustrating a progression from invalid rendering and semantic divergence to asset misplacement and near-perfect reconstruction.



(a) Video duration distribution.

(b) Rubric item count distribution.

Figure 5: Statistics of the WebVR benchmark dataset, showing the distributions of (a) reference video durations (in seconds) and (b) the number of visual rubric items per instance.

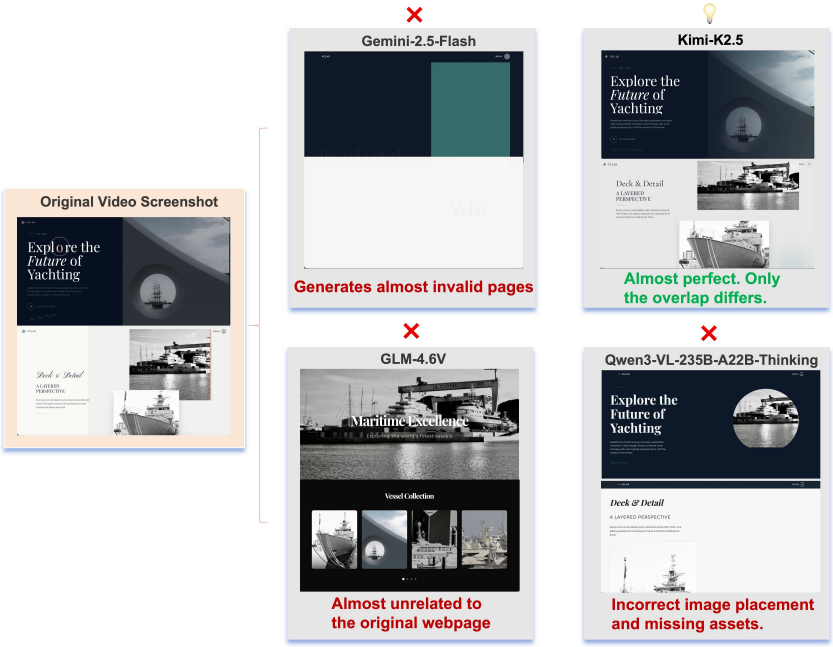


Figure 6: Representative failure cases across four models on the same webpage instance. Gemini-2.5-Flash produces a nearly invalid page with missing content. GLM-4.6V generates a structurally unrelated page that deviates entirely from the reference layout and theme. Qwen3-VL-235B-A22B-Thinking reproduces the overall structure but places images in incorrect locations with wrong references. Kimi-K2.5 achieves near-perfect reconstruction, with only minor inconsistencies in image overlap positioning.