WebWeaver: Style-Aware Webpage Generation with Style Contracts

Runchen Xu[1], **Mengxiao Zhang**[1], **Yikai Gu**[1], **and Jiamou Liu**[1]
[1]School of Computer Science, The University of Auckland, Auckland, NZ
rxu274@aucklanduni.ac.nz, mengxiao.zhang@auckland.ac.nz,
ygu780@aucklanduni.ac.nz, jiamou.liu@auckland.ac.nz

Abstract

We study *style-aware prompt-to-webpage generation*: producing executable webpages that preserve content, functionality, and abstract aesthetic intent. We identify the *design-intent bottleneck* and introduce three components to address it: **Web4Style**, a brief-based diagnostic benchmark; **WebWeaver**, a shared-state framework that constructs and propagates an explicit *style contract*; and **WebJudge**, a structured multimodal evaluator for content, function, and visual aesthetics. Experiments across three backbone models show that WebWeaver improves over direct generation and representative web/code-generation agents in both automatic and human evaluation. These results highlight style contracts as a missing intermediate layer for reliable prompt-to-webpage generation.

1 Introduction

Large language models have made prompt-to-webpage generation increasingly feasible: given a natural-language request, recent code and web-generation agents can produce runnable front-end implementations with HTML, CSS, JavaScript, and interactive components (Lu et al., 2025b,a). Yet a substantial gap remains between generating a page that works and generating one that appears intentionally designed. Professional webpages must not only contain the right elements and interactions; they must also preserve a coherent visual identity across layout, assets, components, and interaction flow.

This gap is especially visible when the input is an abstract design brief rather than a screenshot or detailed mockup. A phrase such as "premium", "playful", or "trustworthy" does not directly specify how the page should be built, but it implies design commitments that should guide the whole page. Existing prompt-to-code systems often leave these implications buried in the prompt, producing pages that satisfy local requirements while feeling visually generic or inconsistent. We call this the *design-intent bottleneck*: abstract aesthetic language must be transformed into explicit, reusable, and checkable design constraints before it can reliably guide webpage generation.

We study *style-aware prompt-to-webpage generation*, where a system must generate an executable webpage from a natural-language design brief while satisfying content, functional, and aesthetic requirements. Unlike screenshot-to-code or design-to-code tasks, the visual target is not given (Beltramelli, 2017; Laurençon et al., 2024; Yun et al., 2024; Si et al., 2025). Unlike functional webpage-generation benchmarks, success cannot be measured only by whether required elements exist or interactions execute (Xiao et al., 2024; Lu et al., 2025b,a). In this setting, the central challenge is to turn underspecified aesthetic cues into consistent design decisions across planning, assets, code, and validation.

We introduce three components around this problem. **Web4Style** is a brief-based diagnostic benchmark derived from professionally designed websites, where each instance abstracts a reference site into a compact professional-style brief without providing a target screenshot. **WebWeaver** is a shared-state generation framework centered on an explicit *style contract*: an intermediate representation that turns grounded aesthetic intent into constraints for page structure, visual assets, implementation, and validation. **WebJudge** is a structured multimodal evaluation protocol that separates visual-style from content and function, evaluates rendered aesthetics with a fixed VisAWI-Pos-inspired rubric (Moshagen and Thielsch, 2010; Perrig et al., 2024), and evaluates content/function through GUI-agent evidence and checklist scoring.

Experiments across Gemini-2.5, DeepSeek-V3.2, and GPT-5.1 evaluate whether explicit style-

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 12283–12319
October 24-29, 2026 ©2026 Association for Computational Linguistics

contract construction and propagation improve generation quality under the same Web4Style briefs and WebJudge protocol. WebWeaver substantially improves style-aware webpage generation over direct LLM generation, OpenHands (Wang et al., 2025), Bolt.diy (StackBlitz Labs, 2024), and WebGen-Agent (Lu et al., 2025a), achieving overall WebJudge win rates of 89.69%, 88.98%, and 90.19% across the three backbones, and an 80.99% top-1 win rate in human preference evaluation.

Our contributions are as follows:

- We identify the *design-intent bottleneck* in prompt-to-webpage generation: abstract aesthetic language must be made operational before it can reliably guide webpage generation.
- We introduce **Web4Style**, a brief-based diagnostic benchmark for evaluating whether systems can translate compact professional-style briefs into coherent executable webpages.
- We propose **WebWeaver**, a shared-state generation framework centered on an explicit *style contract* that carries aesthetic intent through planning, generation, asset creation, and validation.
- We introduce **WebJudge**, a structured multimodal evaluation protocol that jointly assesses functional validity, content alignment, and visual aesthetics.
- We show that WebWeaver substantially improves style-aware webpage generation under Web4Style and WebJudge across multiple backbone models, human preference evaluation, expert-aligned automatic evaluation, and agent-level ablation.

2 Related Work

Website aesthetics and webpage evaluation. Website aesthetics has long been treated as a multidimensional construct rather than a single subjective preference. Prior work distinguishes classical and expressive aesthetics (Lavie and Tractinsky, 2004), and VisAWI operationalizes webpage aesthetics through facets such as simplicity, diversity, colourfulness, and craftsmanship (Moshagen and Thielsch, 2010, 2013; Perrig et al., 2024). These studies motivate our use of structured aesthetic dimensions in WebJudge. Existing webpage-generation evaluations, however, often rely on screenshot similarity, DOM/code similarity, or browser-based functional tests (Yun et al., 2024; Si et al., 2025; Lu et al., 2025b). Such metrics are useful for fidelity and interaction checking, but they do not fully assess whether a generated page realizes the intended visual identity. WebJudge therefore evaluates generated webpages as content artifacts, interactive interfaces, and visual designs, using rendered evidence, GUI-agent traces, and a fixed visual-aesthetics rubric. WebJudge differs from general aesthetic evaluation by assessing both visual quality and alignment with the intended style specified in the original brief. It combines four VisAWI-Pos-inspired dimensions with explicit checks of visual requirements.

Webpage generation from visual targets and textual briefs. Early interface-generation work translated visual inputs into code, including GUI generation from screenshots (Beltramelli, 2017). Recent screenshot-to-code and design-to-code benchmarks extend this direction to realistic webpages and multimodal webpage understanding (Laurençon et al., 2024; Yun et al., 2024; Si et al., 2025). These tasks usually assume that the visual target is already provided. Other work studies front-end generation from natural-language instructions, prototypes, or interaction requirements (Xiao et al., 2024; Sun et al., 2025; Lu et al., 2025b). These settings emphasize runnable code and functional completion, but a functionally correct page may still feel visually generic or inconsistent with the intended design brief. Web4Style targets this gap: systems must infer a visual direction from compact professional-style briefs without access to a target screenshot.

Agentic systems for code and web generation. LLM-based agent systems decompose software tasks into planning, implementation, feedback, and repair (Wu et al., 2023; Hong et al.; Qian et al., 2024; Huang et al., 2023; Yang et al., 2024). Recent web-generation agents also incorporate visual feedback, browser interaction, and GUI-agent testing (Lu et al., 2025a; Li et al., 2026). These systems show the value of agentic decomposition, but professional visual style is rarely maintained as an explicit shared object across generation stages. WebWeaver instead uses multi-agent coordination to construct, propagate, and validate a style contract that preserves aesthetic intent through planning, asset generation, code synthesis, and repair.

3 Task Formulation

We define *style-aware prompt-to-webpage generation* as the task of generating an executable web-

page from a natural-language design brief and optional external knowledge. The target appearance is not directly provided: the system must infer the intended visual direction from language, make that direction operational, and realize it while preserving the requested content and functionality. This setting exposes the *design-intent bottleneck*: aesthetic intent is often expressed abstractly, but generation requires constraints that can guide planning, implementation, and evaluation.

Formally, the input is

$$(R, K), \qquad R = (R_{\text{cont}}, R_{\text{func}}, R_{\text{style}}),$$

where K provides optional external grounding. The component R_{cont} describes what the page should communicate, R_{func} specifies the user actions or interaction requirements, and R_{style} expresses the intended aesthetic direction. The difficulty lies in R_{style}: it is rarely executable as written, and must be translated into a more concrete representation before it can guide page planning, asset creation, code synthesis, or evaluation.

The output is a deployable webpage bundle

$$W = (S, O, V, C),$$

where S is the inferred *style contract*, O the page outline and interaction structure, V the visual assets, and C the executable front-end implementation. The style contract is not user-supplied; it is a system-generated intermediate representation that makes aesthetic intent explicit and checkable across generation and evaluation.

A successful output must satisfy three complementary criteria. *Content alignment* requires the page to communicate the intended message and use external knowledge faithfully when such knowledge is provided. *Functional validity* requires the specified interactions and user actions to be executable. *Visual aesthetics* requires the page to realize the intended style as a coherent rendered experience rather than as isolated decorative choices. These criteria are complementary: a webpage may work correctly while feeling visually generic, look appealing while omitting required content, or preserve a style direction while failing intended interactions.

4 Web4Style Benchmark

Existing webpage-generation benchmarks often assume either that a visual target is already available or that success can be checked through functional requirements. Instead, we design **Web4Style**[1] as a brief-based diagnostic benchmark: it diagnoses whether a system can infer design intent from a compact textual brief and turn that intent into a coherent executable webpage.

Table 1: Comparison of webpage-generation benchmarks. Web4Style targets style abstraction from textual briefs, requiring systems to infer visual direction beyond reconstruction or functional testing.

Benchmark	Input	Func.	Style	Real-site	Brief
Web2Code (Yun et al., 2024)	Img.+Text	✓	×	×	×
Design2Code (Si et al., 2025)	Image	✓	×	✓	×
Interaction2Code (Xiao et al., 2024)	Interactive prototype	✓	×	✓	×
WebGen-Bench (Lu et al., 2025b)	Text	✓	×	✓	✓
Web4Style	Text	✓	✓	✓	✓

Each Web4Style instance abstracts a professional website into a generator-ready brief that specifies its purpose, audience, page flow, visual atmosphere, and supported actions. This preserves high-level design intent while removing the original implementation, brand text, screenshots, and media assets.

We construct Web4Style from professionally curated websites on Awwwards[2]. Each reference site is inspected only transiently for brief construction; we do not retain or redistribute its original code, screenshots, or media assets. We prompt an LLM in the role of a creative director to rewrite the observed design into a compact textual brief that preserves the high-level design intent without copying brand-specific wording or visual assets. Each brief is normalized into a generator-ready instruction beginning with "Build a ...". The full brief-construction prompt is provided in App. A.

We apply format- and content-level checks to ensure that each brief is concrete enough to guide generation while remaining abstract enough not to become a screenshot description. Briefs are revised or removed when they are overly generic, too dependent on proprietary content, or insufficiently informative about the intended experience. We release Web4Style as structured textual briefs with metadata only; raw webpage code, screenshots, and media assets are not redistributed.

Web4Style contains 300 structured briefs across 25 website scenarios. Its purpose is to provide a controlled setting for testing style abstraction from language. As shown in Table 1, Web4Style differs from reconstruction-oriented and function-

[1] https://huggingface.co/datasets/Web4StyleLab/Web4Style.

[2] https://www.awwwards.com/

oriented benchmarks because the system must infer a visual direction from an abstracted brief rather than copy a given target or satisfy only functional requirements. This makes Web4Style a natural testbed for evaluating whether a generator can construct and follow a style contract.

5 WebWeaver Framework

We propose **WebWeaver**, a shared-state generation framework for style-aware prompt-to-webpage generation. Given a design brief R and optional external knowledge K, WebWeaver produces the webpage bundle $W = (S, O, V, C)$, where S is the style contract, O is the page outline, V is the visual asset set, and C is the executable implementation. As shown in Figure 1, the framework first constructs S from the grounded brief, then uses it as a persistent reference for planning, asset generation, code synthesis, and validation. Multi-agent coordination is a means to maintain the style contract, not the contribution itself.

5.1 Shared State

WebWeaver uses shared state to keep design intent stable across generation stages. Starting from $X^{(0)} = (R, K)$, the state is progressively enriched into $X^{(t_{\max})} = (R, K, \hat{R}, G, S, O, V, C, Q)$, where S, O, V, C follow the webpage-bundle notation above, $\hat{R}$ is the normalized request, G the grounded brief, and Q the validation report.

The shared state acts as contract memory. Once S is constructed, downstream agents read it together with the relevant artifacts instead of reinterpreting the raw brief independently. This helps keep page structure, generated assets, and implementation decisions aligned with the same design intent. State-update details and full agent schemas, prompts are provided in App. B and C.

5.2 Style Contract

The *style contract* is WebWeaver's intermediate representation for making abstract aesthetic intent usable. Given the normalized request $\hat{R}$ and grounded brief G, AESTHETIC SPECIFICATION AGENT constructs $S = A_{\text{spec}}(\hat{R}, G)$. Rather than prescribing every pixel, S captures the stable design commitments guiding the page as a whole.

Following Figure 1, we write $S = (S_{\text{sys}}, S_{\text{rhythm}}, S_{\text{interact}}, S_{\text{comp}}, S_{\text{tone}})$, covering the design system, section rhythm, interaction feel, component behavior, and tone or imagery direction. In implementation, these components are expanded into concrete choices such as palette, typography, layout, motion, and asset direction.

A single contract should still allow sections to differ in role. Given a page outline $O = (o_1, \ldots, o_n)$, WebWeaver derives a section-level realization $s_i = \pi(S, o_i)$, so that a hero section, product grid, or contact form can express the same contract in different ways while preserving the page-level visual identity.

5.3 Constructing, Propagating, and Validating the Contract

WebWeaver proceeds through the three stages shown in Figure 1. The main text gives the workflow skeleton, while Appendix B provides the full agent-level workflow, including requirement-aware arbitration and localized repair.

Grounding design intent. The USER AGENT converts the design brief R into a normalized request $\hat{R}$, and the GROUNDING AGENT incorporates optional external knowledge K to produce a grounded brief G. This stage prepares the information needed to construct the style contract.

Constructing and propagating the style contract. The AESTHETIC SPECIFICATION AGENT constructs the style contract S from $\hat{R}$ and G. The PLAN AGENT, GENERATOR AGENT, and ILLUSTRATION AGENT then read S from shared state to produce the page outline O, executable implementation C, and visual assets V. The purpose is to prevent style intent from being independently reinterpreted by each stage.

Validating and repairing contract violations. The VALIDATION AGENT checks the implementation and rendered webpage against the normalized request, page outline, and style contract, producing a validation report Q. The USER AGENT then performs requirement-aware arbitration over Q, $\hat{R}$, and S, deciding whether the webpage is acceptable or whether localized repair is needed. Revision directives are routed to the affected downstream agent, such as the PLAN AGENT, GENERATOR AGENT, or ILLUSTRATION AGENT, rather than restarting the full pipeline.

6 WebJudge Evaluation Protocol

We design **WebJudge**, a structured multimodal evaluation protocol for style-aware prompt-to-webpage generation. Given a generated webpage bundle, WebJudge evaluates whether the rendered

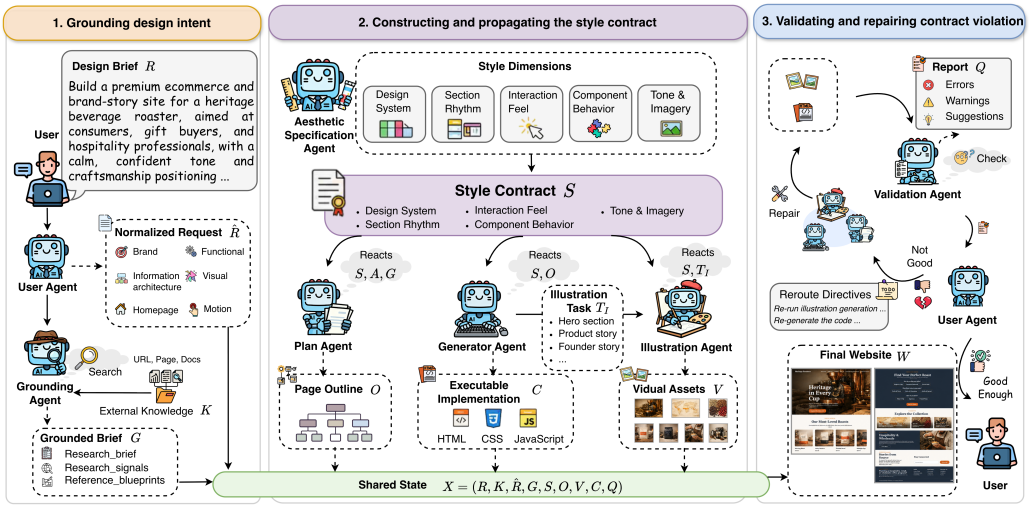


Figure 1: Overview of WebWeaver. The design-intent bottleneck is addressed by constructing a style contract from the grounded brief and propagating it through planning, asset generation, code synthesis, and validation.

page satisfies the brief as a content artifact, an interactive interface, and a visual design. This separation is necessary because a page may function correctly but miss the intended style, or look polished while omitting required content.

WebJudge first separates requirements into visual-style requirements and non-visual requirements. Visual-style requirements are evaluated through the rendered appearance of the page and compared against the intended design direction. Content and functional requirements are evaluated through GUI-agent interaction traces and checklist evidence. This separation prevents stylistic requests from being treated as ordinary functional failures, while also preventing visually attractive pages from receiving high scores when required content or interactions are missing.

Visual aesthetics evaluation. For visual evaluation, WebJudge serves the generated website locally, renders it in a controlled Chromium environment, and captures a full-page screenshot. A multimodal evaluator then applies a fixed VisAWI-Pos-inspired rubric to the rendered page (Moshagen and Thielsch, 2010; Perrig et al., 2024). The rubric produces dimension-level scores and an aggregate visual score, allowing WebJudge to assess visual quality as a structured judgment rather than a single unconstrained preference prompt. The full item-level rubric is provided in App. D. When explicit visual constraints are provided, WebJudge also checks whether they are visibly satisfied.

Functional and content evaluation. For functional validity and content alignment, WebJudge follows a GUI-agent testing paradigm (Lu et al., 2025a). It converts relevant requirements into GUI-testing instructions, interacts with the locally served page, records visible evidence and executed actions, and summarizes the trajectory into a structured report. Functional validity measures whether requested interactions can be exercised and verified. Content alignment measures whether the non-visual information and page structure required by the brief are present. Purely stylistic requirements are excluded from these checklists and handled by the visual rubric.

WebJudge outputs separate scores for visual aesthetics, functional validity, and content alignment, together with the evidence used to support them. In our experiments, these dimensions are reported individually and combined into an overall score for pairwise comparison. Keeping the dimensions separate allows us to distinguish pages that are visually strong but incomplete from pages that are functional but stylistically weak. The exact scoring templates, requirement-separation prompt, GUI-agent instructions, and aggregation details are provided in Appendix D.

WebJudge therefore mirrors the task definition in Sec. 3 by treating webpage generation as a joint content, function, and style problem, allowing the experiments to distinguish between systems that merely produce executable pages from those that also preserve the intended design direction.

7 Experiments and Analysis

Our evaluation asks whether explicit style abstraction and style-contract propagation improve style-

aware prompt-to-webpage generation under a controlled brief-based benchmark. We evaluate WebWeaver on Web4Style using WebJudge, human preference evaluation, expert-aligned WebJudge validation, and ablation analysis. The key comparison is between systems that receive the same design brief but differ in whether they explicitly construct, propagate, and validate a style contract.

Our results should not be read as a universal ranking of all web-development systems under all possible prompt-engineering configurations. Instead, they show that, in a style-aware brief-to-webpage setting, making aesthetic intent explicit and persistent across stages yields substantial gains over direct generation and representative agentic web/code baselines.

We organize the analysis around five questions:

- **RQ1**: Does constructing and propagating a style contract improve style-aware prompt-to-webpage generation?
- **RQ2**: Are the gains limited to visual quality, or do they also preserve content alignment and functional validity?
- **RQ3**: Do humans prefer webpages generated through explicit style-contract propagation?
- **RQ4**: Does WebJudge align with expert judgments as a scalable structured evaluator?
- **RQ5**: Which parts of the style-contract lifecycle contribute most to final quality?

7.1 Experimental Setup

Benchmark. We evaluate all methods on Web4Style. Each instance provides a compact professional-style brief, and each method must generate a complete executable webpage without access to a target screenshot. This setting directly tests whether a system can infer and realize a style contract from language.

Baselines and backbones. We compare WebWeaver against representative direct-generation, code-agent, and web-generation baselines: a vanilla LLM baseline, OpenHands (et al., 2024), Bolt.diy (StackBlitz Labs, 2024), and WebGen-Agent (Lu et al., 2025a). The vanilla baseline directly generates the webpage from the brief without constructing a persistent style contract. OpenHands and Bolt.diy represent general-purpose code or web-development agents, while WebGen-Agent represents an agentic webpage-generation system. These baselines may still express style from prompts, but they are not organized around explicit style-contract construction, propagation, and validation. We instantiate LLM-based methods with Gemini-2.5, DeepSeek-V3.2, and GPT-5.1 to test whether the observed effects are robust across backbones.

Metrics and evaluation protocols. We evaluate generated webpages using WebJudge (Sec. 6), which separately assesses functional validity, content alignment, and visual aesthetics. We use GPT-4o-mini as the fixed evaluator model in WebJudge for all methods and backbone settings to ensure a consistent evaluation protocol. Scores are derived from requirement-specific checklists, GUI-agent evidence, and rendered-page aesthetic evaluation. We report dimension-level and overall scores for pairwise comparison, with templates and aggregation details in App. D. All pairwise results are aggregated over 125 briefs, with each method generated once per task.

Human preference study. To assess whether WebJudge improvements are perceptible to users, we conduct a human preference study. Participants compare anonymized rendered webpages for the same Web4Style brief, with access to the brief but not method names. We recruit 120 participants to evaluate 125 briefs; each participant ranks two randomized candidate sets, and each brief receives multiple judgments. We report Top-1 win rate, mean rank with standard deviation, and 95% bootstrap confidence intervals. The questionnaire and further details are provided in App. E.

WebJudge reliability. We also examine whether WebJudge rankings align with expert judgments. We collect blind comparative judgments from 20 experts on 125 candidate sets, each containing webpages generated by different systems for the same brief. Experts rate the webpages on the VisAWI-Pos dimensions and overall preference, independently of the WebJudge pipeline and without access to intermediate style contracts or agent states. We measure agreement using Spearman's ρ, Kendall's τ, Top-1 Accuracy, and nDCG@3. We also run a rubric-free VLM check as an additional validation; see App. E.2.

Further reproducibility and execution details are provided in App. E.3.

7.2 Experimental Analysis

Automatic evaluation. Tab. 2 shows WebWeaver achieves the highest overall win rate across all backbones: 89.69% with Gemini-2.5, 88.98% with DeepSeek-V3.2, and 90.19% with GPT-5.1.

Table 2: Average pairwise WebJudge win rates (%) on Web4Style across three backbones. Func. and Cont. denote functional validity and content alignment, respectively. Style Avg. is the unweighted mean of Simplicity, Diversity, Colourfulness, and Craftsmanship. Bold marks the best result per backbone.

Backbone	Method	Func.	Cont.	Visual Aesthetics				Style Avg.	Overall
				Simp.	Div.	Color.	Craft.		
Gemini-2.5	LLM	54.43%	85.71%	53.57%	43.45%	44.05%	46.43%	46.88%	62.34%
	OpenHands	16.71%	88.14%	48.81%	36.90%	36.31%	36.31%	39.58%	48.14%
	Bolt.diy	26.57%	69.29%	37.50%	50.60%	50.60%	48.81%	46.88%	47.58%
	WebGen-Agent	23.86%	57.15%	33.93%	22.62%	24.40%	26.79%	26.94%	36.32%
	WebWeaver	**85.50%**	**93.83%**	**76.19%**	**96.43%**	**94.64%**	**91.67%**	**89.73%**	**89.69%**
DeepSeek-V3.2	LLM	58.71%	86.71%	65.28%	59.72%	45.83%	59.72%	57.64%	67.69%
	OpenHands	24.71%	85.47%	33.33%	36.11%	50.00%	37.50%	39.24%	49.81%
	Bolt.diy	27.83%	86.20%	29.17%	15.23%	15.28%	15.28%	18.99%	44.34%
	WebGen-Agent	23.86%	57.15%	26.38%	41.67%	48.61%	38.89%	38.89%	39.97%
	WebWeaver	**76.75%**	**94.71%**	**95.83%**	**97.22%**	**90.28%**	**98.61%**	**95.49%**	**88.98%**
GPT-5.1	LLM	62.14%	89.23%	32.14%	35.71%	28.57%	35.71%	33.03%	61.47%
	OpenHands	21.36%	87.54%	60.71%	67.86%	71.43%	64.29%	66.07%	58.32%
	Bolt.diy	31.82%	70.68%	28.57%	25.00%	28.57%	21.43%	25.89%	42.80%
	WebGen-Agent	27.45%	60.92%	32.14%	39.29%	53.57%	35.71%	40.18%	42.85%
	WebWeaver	**81.76%**	**96.85%**	**96.43%**	**85.71%**	**92.86%**	**92.86%**	**91.97%**	**90.19%**

Table 3: Human preference evaluation on Web4Style. Top-1 and mean rank measure holistic preference over anonymized rendered webpages; 95% confidence intervals are computed by bootstrap.

Method	Top-1 (95% CI) ↑	MR (mean±sd) / 95% CI ↓
WebWeaver	**80.99** [73.75, 86.59]	**1.40±0.95** / [1.24, 1.56]
Bolt.diy	8.45 [4.90, 14.19]	2.94±1.17 / [2.75, 3.14]
OpenHands	5.63 [2.88, 10.72]	3.67±1.22 / [3.47, 3.87]
LLM	3.52 [1.51, 7.98]	3.26±1.18 / [3.07, 3.46]
WebGen-Agent	1.41 [0.39, 4.99]	3.73±1.13 / [3.54, 3.91]

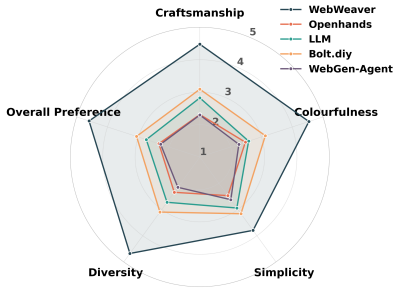


Figure 2: Human preference results. The chart summarizes perceived quality across evaluation dimensions.

Table 4: Ablation of the WebWeaver style-contract lifecycle using Gemini-2.5. Each variant removes one stage or agent.

Metric	Full	w/o Ground.	w/o Spec.	w/o Plan	w/o Illus.	w/o Val.
Func.	**85.50**	76.71	43.40	17.83	54.33	57.86
Cont.	**93.83**	96.43	89.55	93.83	86.50	94.92
Simp.	76.19	66.58	69.45	**77.18**	54.62	52.19
Div.	**96.43**	40.12	43.21	59.87	44.78	71.62
Color.	**94.64**	36.28	47.35	65.74	48.91	75.30
Craft.	**91.67**	41.23	55.89	60.32	43.56	73.46
Style Avg.	**89.73**	46.05	53.98	65.78	47.97	68.14
Overall	**89.69**	73.06	62.31	59.15	62.93	73.64

These consistent gains suggest that constructing and propagating an explicit style contract improves style-aware webpage generation.

The strongest gains are in visual aesthetics: WebWeaver achieves the best Style Avg. under all backbones (89.73%, 95.49%, and 91.97%) and leads across all VisAWI-Pos dimensions. This suggests that the style contract improves visual quality across multiple aesthetic aspects, not only a holistic preference score.

The gains also extend beyond visual quality. WebWeaver achieves the highest functional validity and content alignment under all three backbones, suggesting that grounding, contract-based planning, and rendered-output validation improve visual design without sacrificing required content or executable interactions.

We further examine representative functional and content failure cases to better understand the remaining limitations; see App. H.

Human preference evaluation. Human evaluation provides external evidence that the style-contract pipeline improves perceived webpage quality beyond WebJudge scores. Tab. 3 shows WebWeaver achieves an 80.99% Top-1 win rate and the best mean rank of 1.40 ± 0.95, while all baselines have Top-1 win rates below 10% and mean ranks above 2.9. This shows a clear human preference for explicit style-contract propagation.

Figure 2 shows that the preference extends beyond holistic quality: WebWeaver is perceived strongly across aesthetic aspects, especially craftsmanship and diversity. This suggests that style-contract propagation yields pages that feel more intentionally composed, while its gains in simplicity and colourfulness indicate expressive design without sacrificing clean organization.

Ablation analysis. We ablate the style-contract lifecycle by removing one stage/agent at a time, while keeping input briefs, backbone, and WebJudge fixed. Removed outputs are replaced by the closest available fallback, allowing us to test

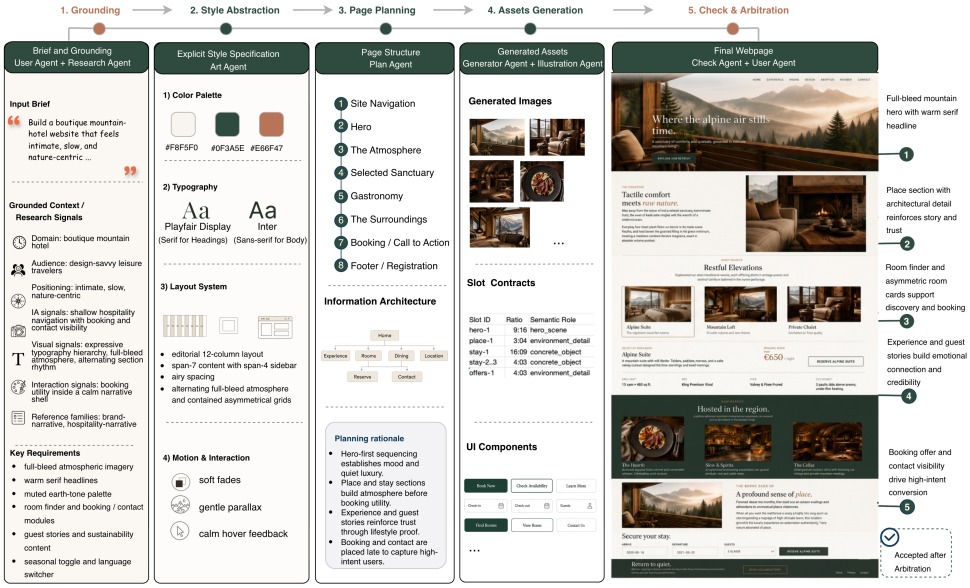


Figure 3: Case study from Web4Style sample `000061`: WebWeaver propagates a boutique mountain-hotel style contract into page rhythm, visual assets, code, and booking interactions.

whether the style contract functions merely as a static prompt artifact or requires grounding, construction, propagation, and validation.

Table 4 shows that each stage supports different task dimensions. Removing the GROUNDING AGENT sharply reduces visual richness and craftsmanship, while removing the AESTHETIC SPECIFICATION AGENT causes broad drops, confirming that explicitly constructing S is central. Removing the PLAN AGENT yields the largest functional decline, showing that the style contract must guide page structure and interaction flow, not only surface appearance. The slight simplicity increase without the PLAN AGENT likely reflects structurally simpler but less complete pages. Removing the ILLUSTRATION AGENT mainly weakens visual polish, and removing the VALIDATION AGENT lowers overall quality, showing the value of checking and repairing contract violations. Overall, WebWeaver's gains do not come from agent count alone, but from a lifecycle that grounds aesthetic intent, makes it explicit as a style contract, propagates it into structure, code, and assets, and validates the rendered result.

WebJudge reliability. Table 5 shows strong alignment between WebJudge and expert judgment across VisAWI-Pos dimensions and overall preference. Spearman's ρ ranges from 0.737 to 0.774, Kendall's τ remains high, and all nDCG@3 scores exceed 0.93, indicating that WebJudge largely preserves expert rankings and places strong candidates near the top. The lower Top-1 Accuracy for simplicity suggests that minimalist layout quality may require finer-grained judgment over spacing, hierarchy, and balance. The rubric-free VLM check provides complementary evidence for the visual gains; see App. E.2.

Table 5: Alignment between WebJudge and expert judgments. The results test whether WebJudge can serve as a scalable structured evaluator for visual-aesthetic ranking and overall preference.

Metric	Simp.	Div.	Colour.	Craft.	Overall Pref.
Spearman's $\rho \uparrow$	0.737	0.774	0.753	0.763	0.763
Kendall's $\tau \uparrow$	0.716	0.684	0.642	0.663	0.674
Top-1 Acc. $\uparrow$	0.632	0.895	0.842	0.789	0.842
nDCG@3 $\uparrow$	0.933	0.938	0.930	0.932	0.937

Case study. We present Web4Style sample `000061` as a qualitative example of the style-contract lifecycle. The brief describes a boutique mountain-hotel website with an intimate, slow, nature-centered atmosphere and lightweight booking interactions; the full brief appears in App. A.

WebWeaver grounds the brief, constructs a quiet editorial style contract, propagates it into a page outline from ambience and rooms to surroundings and booking, and realizes it through code and assets before validation. As shown in Figure 3, the page maintains a coherent visual identity while varying across sections, illustrating how the style contract supports both consistency and progression rather than mechanically repeating one visual pattern.

Additional case studies are provided in App. G.

Generalization on WebGen-Bench. To assess generalization beyond Web4Style and WebJudge, we additionally evaluate all methods on WebGen-Bench (Lu et al., 2025b) under their original protocols. WebWeaver achieves the best performance on both functional and visual metrics; detailed setup and results are provided in App. F.

8 Conclusion

We introduced Web4Style, WebWeaver, and WebJudge for style-aware prompt-to-webpage generation. The core idea is the style contract: a system-generated representation that turns abstract design intent into a persistent, shareable, and evaluable guide for generation. Web4Style tests this capability, WebWeaver constructs and propagates the contract, and WebJudge evaluates the resulting webpages across content, function, and visual design. Experiments, human evaluation, WebJudge validation, and ablations show that style-contract propagation improves generation quality.

9 Limitations

Web4Style is designed as a brief-based diagnostic benchmark, not as a comprehensive model of all website requests. It is derived from curated professional websites and may therefore overrepresent visually expressive, design-forward web genres. Its briefs are LLM-assisted abstractions rather than naturally occurring user prompts, so Web4Style should be interpreted as a controlled testbed for style abstraction from language rather than a large-scale pretraining corpus or a complete sample of real-world web-design needs.

WebWeaver evaluates the effect of explicit style-contract construction and propagation for front-end webpage generation. The generated artifacts include executable front-end code, visual assets, layout, and interactions, but we do not comprehensively evaluate backend integration, database connectivity, authentication, security, deployment robustness, long-term maintainability, SEO, or production-level engineering constraints. The paper therefore evaluates style-aware front-end webpage generation, not complete production web-application development.

WebJudge provides a scalable structured evaluation protocol, but it is not a substitute for full expert design review. Although our reliability study shows strong alignment with expert judgments, automatic evaluation may still miss subtle issues involving accessibility, mobile responsiveness, cultural design conventions, semantic HTML quality, performance, or long-term user experience. Future versions of WebJudge could incorporate additional criteria and task-specific rubrics.

Finally, our comparisons are conducted under the Web4Style/WebJudge setting. Stronger task-specific prompt engineering, additional repair loops, or tool-specific adaptation may improve baseline systems and narrow the observed gap. Our results should therefore be interpreted as evidence for the value of explicit style contracts under a controlled style-aware generation setting, rather than as a universal ranking of all web-development tools.

10 Ethical Considerations

The primary ethical considerations concern the use of publicly accessible web content, the abstraction of professional design intent, and the handling of human evaluation data. Web4Style is constructed from publicly accessible professional websites for research purposes, but we release only structured textual briefs and metadata. We do not redistribute original webpage code, screenshots, layouts, videos, or media assets. The benchmark is intended to study style abstraction from language rather than reproduce copyrighted website implementations or brand-specific creative assets.

Human evaluation data are used only for aggregate research analysis and are not used for participant profiling or model training. Candidate webpages are anonymized with respect to generation method during evaluation. We follow standard ethical guidelines for human-subject evaluation and data handling.

References

Tony Beltramelli. 2017. pix2code: Generating code from a graphical user interface screenshot. *arXiv preprint arXiv:1705.07962*.

Wang et al. 2024. Openhands. Https://github.com/OpenHands/OpenHands.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. Metagpt: Meta programming for a multi-

agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Dong Huang, Qingwen Bu, Jie M. Zhang, Michael Luck, Yuhao Qing, and Heming Cui. 2023. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*.

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*.

Talia Lavie and Noam Tractinsky. 2004. Assessing dimensions of perceived visual aesthetics of web sites. *International Journal of Human-Computer Studies*, 60(3):269–298.

Yan Li, Zezi Zeng, Yifan Yang, Yuqing Yang, Ning Liao, Weiwei Guo, Lili Qiu, Mingxi Cheng, Qi Dai, Zhendong Wang, Zhengyuan Yang, Xue Yang, Ji Li, Lijuan Wang, and Chong Luo. 2026. MM-WebAgent: A hierarchical multimodal web agent for webpage generation. *arXiv preprint arXiv:2604.15309*.

Zimu Lu, Houxing Ren, Yunqiao Yang, Ke Wang, Zhuofan Zong, Junting Pan, Mingjie Zhan, and Hongsheng Li. 2025a. Webgen-agent: Enhancing interactive website generation with multi-level feedback and step-level reinforcement learning. *arXiv preprint arXiv:2509.22644*.

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

Morten Moshagen and Meinald Thielsch. 2013. A short version of the visual aesthetics of websites inventory. *Behaviour & Information Technology*, 32(12):1305–1311.

Morten Moshagen and Meinald T. Thielsch. 2010. Facets of visual aesthetics. *International Journal of Human-Computer Studies*, 68(10):689–709.

Sebastian AC Perrig, Nick von Felten, Marimo Honda, Klaus Opwis, and Florian Brühlmann. 2024. Development and validation of a positive-item version of the visual aesthetics of websites inventory: The visawi-pos. *International Journal of Human–Computer Interaction*, 40(20):6622–6646.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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.

StackBlitz Labs. 2024. Bolt.diy. Https://github.com/stackblitz-labs/bolt.diy.

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*.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2025. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pages 65882–65919.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, Chi Wang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, and Doug Burger. 2023. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Jingyu Xiao, Yuxuan Wan, Yintong Huo, Zhiyao Xu, and Chi-Wing Fu. 2024. Interaction2code: How far are we from automatic interactive webpage generation? *arXiv preprint arXiv:2411.03292*.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, Timothy Baldwin, Zhengzhong Liu, Eric P. Xing, Xiaodan Liang, and Zhiqiang Shen. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal LLMs. In *Advances in Neural Information Processing Systems*, volume 37.

APPENDIX

A Web4Style Construction and Example Brief

This section provides additional details on the construction of Web4Style. As described in Section 4, Web4Style is a brief-based diagnostic benchmark rather than a large-scale pretraining corpus. Its purpose is to test whether a system can infer a style contract from compact professional-style briefs and realize that contract in an executable webpage.

Source and abstraction. Web4Style is derived from professionally curated websites on Awwwards, used only as transient references. We do not retain or redistribute original webpage code, screenshots, videos, layouts, or media assets. Instead, each reference site is abstracted into a generator-ready design brief that preserves high-level design intent while removing brand-specific wording and protected creative assets.

Overview. Table 6 summarizes the source, scale, construction pipeline, and release format of Web4Style.

Table 6: Overview of Web4Style as a brief-based diagnostic benchmark.

Aspect	Description
Source	Professionally curated websites from Awwwards, used only as transient references.
Size	300 structured design briefs.
Categories	25 categories, each containing 12 examples, spanning architecture, art, business, education, ecommerce, fashion, hospitality, luxury, media, mobile apps, photography, real estate, sports, startups, and technology.
Pipeline	Reference selection, transient webpage inspection, LLM-assisted brief abstraction, normalization, and quality checking.
Filtering	Duplicate, inaccessible, brand-dependent, overly dynamic, or insufficiently informative sites were excluded.
Release	Structured textual briefs and metadata. Raw HTML, CSS, JavaScript, screenshots, images, videos, and website assets are not redistributed.

Brief construction and quality control. We prompt an LLM in the role of a creative director to rewrite the observed design into a compact professional-style brief. Each brief is normalized into a generator-ready instruction beginning with "Build a ...". The prompt is designed to abstract high-level design intent rather than reproduce the original website, its code, layout, screenshots, or brand-specific assets. We apply format- and content-level checks to ensure that each brief is concrete enough to guide generation while remaining abstract enough not to become a reconstruction prompt. Before brief construction, we excluded duplicate, inaccessible, brand-dependent, overly dynamic, or insufficiently informative reference sites. Briefs were then revised or removed when they were overly generic, reliant on proprietary content, or unclear about the intended experience.

Representative examples cover diverse scenarios and style regimes, including an interior-design portfolio (000001), luxury yacht brokerage (000011), heritage e-commerce site (000051), boutique hotel (000061), and digital product studio (000081). Details of these example briefs are provided in App. G.

Example brief. We show one Web4Style sample below. This example illustrates the design-intent bottleneck: the brief gives a compact professional-style direction, but a generator must still infer a style contract before planning, implementation, and evaluation.

Sample `000061`.

> Build a boutique mountain-hotel website that feels intimate, slow, and nature-centric, targeting design-savvy leisure travelers seeking authentic, small-scale stays; use a shallow IA with primary nav for Hotel, Rooms, Culinary, Nature & Activities, Wellness, Offers, Story, and Contact/Booking, plus language switcher and seasonal Summer/Winter toggles. The homepage should scroll as a narrative (hero promise, location, rooms, cuisine, activities, wellness, offers, testimonials, sustainability, contact), with each block full bleed imagery, restrained copy, and a clear CTA, revealed via soft fades and parallax drifts. Use a generous typographic scale (warm serif for headlines, clean sans for body), airy spacing, and a muted earth tone palette with high-contrast text on off-white or deep green panels, minimal shadows, and subtle micro-interactions on hover and anchor scroll; core modules include room finder, offer high-

lights, activity teasers, review carousel, newsletter signup, and simple booking/contact forms, while excluding e-commerce, complex loyalty systems, or social feeds beyond basic icons.

Professional-Style Brief Construction Prompt. We use the following prompt to construct compact design briefs from transiently inspected reference websites. The prompt text is preserved verbatim for reproducibility.

```
BRIEF_EXTRACTION_PROMPT = """You are extracting a website-
      generation brief from a reference website.
Write like a creative director handing off to a designer
      and frontend developer, focusing on structure, pacing
      , and visual intent rather than surface description.

Analyze this website:{URL}

Write a single cohesive, generator-ready brief in English.

Opening requirement:
- The output MUST begin with a sentence starting exactly
      with:
  "Build a ..."

The brief must capture:

- what the site is
- site type and primary audience
- emotional tone and brand positioning
- navigation structure and information architecture depth
- homepage flow and section sequencing
- media usage and visual hierarchy
- typography scale, contrast, and spacing system
- color system, contrast strategy, and overall visual mood
- material feel and surface treatment
- interaction and motion style
- functional scope and core modules
- explicitly excluded features where relevant
- primary conversion goal and user actions

Requirements:

- Make it concrete, implementation-ready, and adaptable
      across industries
- Preserve pacing, hierarchy, and editorial rhythm with
      clear section progression
- Emphasize key modules and interaction patterns
- Define how content reveals and transitions across scroll
- Do not copy original text
- Do not mention the original brand

STRICT LENGTH REQUIREMENT:

- The output MUST be between 80 and 120 words.
- Single paragraph only
- No bullet points or section labels

Density constraint:

- Every sentence must include multiple design decisions
- Merge ideas instead of listing them
- Avoid filler or generic phrasing
"""
```

B Full WebWeaver Workflow

This section gives the full procedural workflow of WebWeaver. The main paper presents the compact workflow skeleton; here we describe the agent-level procedure, including shared state, style-contract construction, propagation, validation, requirement-aware arbitration, and localized repair.

B.1 Overview

Given a design brief R and optional external knowledge K, WebWeaver generates a webpage bundle

$$W = (S, O, V, C),$$

where S is the style contract, O is the page outline and interaction structure, V is the set of visual assets, and C is the executable front-end implementation. The workflow is organized around a shared state

$$X = (R, K, \hat{R}, G, S, O, V, C, Q),$$

where $\hat{R}$ is the normalized request, G is the grounded brief, and Q is the validation report.

The shared state makes intermediate artifacts persistent and inspectable. Rather than letting each agent reinterpret the raw design brief independently, WebWeaver stores the style contract S and related artifacts so that downstream agents can read and apply a common representation of aesthetic intent.

The workflow proceeds through three stages:

1. **Grounding design intent**: the raw design brief and optional knowledge are converted into a normalized request and grounded brief.

2. **Constructing and propagating the style contract**: the grounded brief is transformed into a style contract, which then conditions page planning, code synthesis, and asset generation.

3. **Validating and repairing contract violations**: the generated webpage is checked against the normalized request, page outline, and style contract; localized revision directives are routed to affected downstream agents.

B.2 Shared State and Agents

Let $X^{(t)}$ denote the shared state at stage t. Initially,

$$X^{(0)} = (R, K).$$

Each agent is modeled as a state transformation

$$X^{(t+1)} = A^{(t)}(X^{(t)}),$$

where $A^{(t)}$ reads selected artifacts from the current state and writes one or more new artifacts. The final state has the form

$$X^{(t_{\max})} = (R, K, \hat{R}, G, S, O, V, C, Q).$$

The main agents are:

- USER AGENT: normalizes the raw design brief into $\hat{R}$, and later performs requirement-aware arbitration during validation.
- GROUNDING AGENT: incorporates optional external knowledge and produces the grounded brief G.
- AESTHETIC SPECIFICATION AGENT: constructs the style contract S.
- PLAN AGENT: produces the page outline and interaction structure O.
- GENERATOR AGENT: produces executable front-end code C and auxiliary illustration tasks T_I.
- ILLUSTRATION AGENT: realizes visual assets V under the style contract.
- VALIDATION AGENT: produces the validation report Q.

The USER AGENT appears twice. At the start, it turns the raw design brief into a structured normalized request. At the end, it uses this normalized request to arbitrate whether the validation report indicates an acceptable webpage or whether localized repair is needed. This preserves the original user intent throughout the workflow rather than treating it as a one-time input.

B.3 Stage 1: Grounding Design Intent

Requirement normalization. The USER AGENT reads the raw design brief R and produces a normalized request:

$$\hat{R} = A_{\text{req}}(R).$$

The normalized request separates the communicative goal, functional requirements, content expectations, and abstract style intent. This step reduces ambiguity before style-contract construction. For example, a phrase such as "premium hospitality website" may imply both content goals and style expectations; the normalized request records these expectations in a form that downstream agents can use.

Context grounding. The GROUNDING AGENT incorporates optional external knowledge K and produces a grounded brief:

$$G = A_{\text{ground}}(\hat{R}, K).$$

The grounded brief adds domain context, audience expectations, brand or product cues, and interaction constraints. This stage prepares the evidence from which the style contract can be constructed. It does not yet produce S; rather, it clarifies the design problem that S must operationalize.

B.4 Stage 2: Constructing and Propagating the Style Contract

Style-contract construction. The AESTHETIC SPECIFICATION AGENT reads the normalized request and grounded brief, then constructs the style contract:

$$S = A_{\text{spec}}(\hat{R}, G).$$

The style contract is the intermediate representation between abstract style language and executable webpage code. It turns R_{style} into constraints that can be shared, instantiated, and checked.

Following the main paper, we organize the style contract as

$$S = (S_{\text{sys}}, S_{\text{rhythm}}, S_{\text{interact}}, S_{\text{comp}}, S_{\text{tone}}),$$

where S_{sys} describes the design system, S_{rhythm} describes section rhythm, S_{interact} describes interaction feel, S_{comp} describes component behavior, and S_{tone} describes tone and imagery. In implementation, these components are expanded into concrete choices such as palette, typography, layout behavior, motion, component treatment, and asset direction.

Page planning. The PLAN AGENT reads the style contract together with the normalized request and grounded brief, then produces the page outline:

$$O = A_{\text{plan}}(S, \hat{R}, G).$$

The page outline defines the sequence of sections, the role of each section, the interaction flow, and the relationship between content blocks. The style contract influences the outline before code is written. This is important because many style failures originate in page structure, density, and section sequencing rather than in surface styling alone.

Section-level realization. Given a page outline

$$O = (o_1, \ldots, o_n),$$

WebWeaver derives section-level realizations

$$s_i = \pi(S, o_i),$$

where s_i adapts the global style contract to the role of section o_i. This allows the webpage to preserve a shared visual identity while letting different sections play different communicative roles.

Code synthesis and illustration tasks. The GENERATOR AGENT reads the style contract and page outline, then produces executable implementation and auxiliary illustration tasks:

$$(C, T_I) = A_{\text{gen}}(S, O).$$

Here, C contains the front-end implementation and T_I describes the visual assets needed by the page. The generator is conditioned on S, so the implementation should not treat style as a loose afterthought.

Asset generation. The ILLUSTRATION AGENT reads the illustration tasks and style contract, then produces visual assets:

$$V = A_{\text{ill}}(T_I, S).$$

The visual assets should follow the same style contract as the page layout and implementation. This prevents the imagery from being generated independently of the overall design direction.

B.5 Stage 3: Validating and Repairing Contract Violations

Validation report. The VALIDATION AGENT reads the generated artifacts and produces a validation report:

$$Q = A_{\text{val}}(\hat{R}, G, S, O, V, C).$$

We write

$$Q = (E, \Omega, \Delta),$$

where E contains blocking errors, Ω contains non-blocking warnings, and Δ contains candidate revision directives. The validation process combines implementation-level inspection with rendered review of the webpage. It checks whether required functions are present, whether content and structure match the brief, and whether the rendered page remains faithful to the style contract.

Requirement-aware arbitration. After validation, the USER AGENT performs requirement-aware arbitration:

$$(y, \rho) = A^{\text{arb}}_{\text{user}}(\hat{R}, S, Q),$$

where $y \in \{0, 1\}$ indicates acceptance and ρ contains localized repair instructions. The VALIDATION AGENT provides evidence about the generated webpage; the USER AGENT interprets that evidence relative to the normalized request and style contract. This separation is useful because not every warning requires repair, and not every unusual visual choice is a failure.

Localized repair. If $y = 1$, the current webpage bundle is accepted. If $y = 0$, the repair instructions ρ are routed to affected downstream agents. Structural failures are routed to the PLAN AGENT; implementation failures are routed to the GENERATOR AGENT; asset or imagery mismatches are routed to the ILLUSTRATION AGENT; and ambiguity in the style contract may be routed back to the AESTHETIC SPECIFICATION AGENT. A repaired artifact is written back to the shared state, after which validation can be run again.

The loop can be summarized as

$$Q \to (y, \rho) \to X^{(t+1)} \to Q',$$

where Q' is the validation report after repair. This loop lets WebWeaver treat style failures as localized contract violations rather than restarting the entire pipeline.

B.6 Summary of Agent Inputs and Outputs

Table 7 summarizes the main agents, their inputs, and their outputs.

C Agent Prompts and Schemas

This section provides the detailed input–output schemas and prompt templates used by WebWeaver. The surrounding prose uses the figure-aligned terminology from Section 5. The prompt boxes are preserved verbatim for reproducibility; some internal prompt wording may use implementation-oriented phrasing, but the formal module names in this paper are USER AGENT, GROUNDING AGENT, AESTHETIC SPECIFICATION AGENT, PLAN AGENT, GENERATOR AGENT, ILLUSTRATION AGENT, VALIDATION AGENT, and requirement-aware arbitration.

Table 7: Summary of WebWeaver agent inputs and outputs. The style contract S is the central intermediate representation propagated through planning, implementation, asset generation, validation, and repair.

Agent	Input	Output
USER AGENT	Raw design brief R	Normalized request $\hat{R}$
GROUNDING AGENT	Normalized request $\hat{R}$, optional external knowledge K	Grounded brief G
AESTHETIC SPECIFICATION AGENT	Normalized request $\hat{R}$, grounded brief G	Style contract S
PLAN AGENT	Style contract S, normalized request $\hat{R}$, grounded brief G	Page outline and interaction structure O
GENERATOR AGENT	Style contract S, page outline O	Executable implementation C, illustration tasks T_I
ILLUSTRATION AGENT	Illustration tasks T_I, style contract S	Visual assets V
VALIDATION AGENT	$\hat{R}, G, S, O, V, C$	Validation report $Q = (E, \Omega, \Delta)$
USER AGENT *(arbitration role)*	Normalized request $\hat{R}$, style contract S, validation report Q	Acceptance decision y, repair instructions ρ

C.1 Overview

Table 8 summarizes the role, input, and output of each agent.

C.2 Requirement Agent

The USER AGENT converts the raw design brief R into a structured normalized request $\hat{R}$. This step records the user's communicative, functional, content, and style intent in a stable form that later stages can read from shared state.

Normalized request $\hat{R}$:
"brand": {"domain", "positioning", "audience", "values"},
"ia": {"structure", "navigation"},
"visual": {"typography", "palette", "textures"},
"homepage": [...],
"motion": {"transitions", "scrolling"},
"functional": {"required", "excluded"},
"conversion": {"strategy", "tactics"}

Prompt.

```
## Purpose
Normalize the raw user brief into a structured request,
      expand underspecified briefs when needed, and provide
       stable reroute policy for leader-mode retries.

## Inputs
- Raw user prompt or normalized brief
- Existing structured request when present
- Repair/check findings for reroute

## Output Contract
- `user_request`
- `user_request_structured`
- Stable reroute rationale

## Prompt Blocks
### normalize_brief
```text
You are extracting a website-generation brief from sparse
      or detailed user intent.
Return STRICT JSON only. No markdown. No explanations.

Use the exact top-level shape:
{
  "brand": {},
  "ia": {},
  "homepage": [],
  "visual": {},
  "motion": {},
  "functional": {},
  "conversion": {},
  "brief": ""
}

Rules:
- Keep the structure compact and implementation-ready.
- Fill dictionaries and homepage sequence with concrete
      decisions.
- `homepage` must describe ordered section progression.
- `functional` must include required and excluded
      capability intent where available.
- `brief` must be one English paragraph, 150-200 words,
      cohesive and generator-ready.
- Align the paragraph with the structured fields.
- Do not invent brand names or unsupported facts.
- Keep scope realistic for browser-only implementation.
```

expand_brief
```text
You are expanding sparse website intent into a dense,
      generator-ready brief.
Write like a creative director handing off to a designer
      and frontend developer.

Capture:
- site type and audience
- emotional tone and brand position
- navigation and information architecture depth
- homepage flow and section sequencing
- media usage and visual hierarchy
- typography, color, spacing, and material feel
- motion and interaction style
- functional scope, exclusions, and conversion goal

Rules:
- 150-200 words total
- single paragraph only
- no bullet points or labels
- every sentence should carry multiple concrete design
      decisions
- keep it implementation-ready and browser-realistic
- do not invent specific brand facts
```

leader_reroute
```text
You are the leader policy for retry routing.

Rules:
- Preserve writable HTML artifacts whenever they exist.
- Prefer rerouting only the earliest stage that can fix the
       fatal issue.
```
```

Table 8: Overview of WebWeaver agent schemas.

Agent	Input	Output
USER AGENT	R	$\hat{R}$
GROUNDING AGENT	$(\hat{R}, K)$	G
AESTHETIC SPECIFICATION AGENT	$(\hat{R}, G)$	S
PLAN AGENT	$(S, \hat{R}, G)$	O
GENERATOR AGENT	(S, O)	(C, T_I)
ILLUSTRATION AGENT	(T_I, S)	V
VALIDATION AGENT	$(\hat{R}, G, S, O, V, C)$	Q
USER AGENT *(arbitration)*	$(\hat{R}, S, Q)$	(y, ρ)

```
- Explicit ecommerce, booking, dashboard, and tool intent
      outrank portfolio or editorial style bias.
- Repeated fatal signatures should terminate with preserved
       artifacts in best-effort mode.
- Produce stable, machine-readable reasoning and avoid
      stage thrash.
```

C.3 Grounding Agent

The GROUNDING AGENT grounds the normalized request $\hat{R}$ with optional external knowledge K. Its role is to enrich the request with domain context, audience expectations, interaction constraints, and relevant design signals, producing the grounded brief G.

Grounded brief G:
"research_brief": {"site_category", "audience", "content_focus", "conversion_intent"},
"research_signals": {"stylistic_cues", "interaction_expectations", "design_risk_flags"},
"reference_blueprints": [{"name", "layout_pattern", "component_direction", "applicability"}]

Prompt.

```
## Purpose
Ground the request in design context, classify the site
      correctly, and synthesize a research brief without
      letting benchmark style bias overwrite explicit
      product intent.

## Inputs
- Normalized user request
- Structured request and implicit constraints
- Optional benchmark context
- Optional fetched snippets and reference candidates

## Output Contract
- `research_brief`
- `experience_mode`
- `reference_signals`
- `design_risk_flags`
- `reference_evidence`
- `reference_blueprints`
- `selected_blueprint`
- Reference summary aligned with the request type

## Prompt Blocks
### synthesis
```text
You are synthesizing a concise design/research brief for a
      multi-agent web generation pipeline.

Rules:
- Research is a gate, not a decorative add-on. Do not
      invent layout structure before evidence has been
      reviewed.
- Preserve the explicit functional site type from the
      request.
- Benchmark context may refine tone and reference choice,
      but must not overwrite the core product type.
- Treat successfully fetched curated references as valid
      evidence, but distinguish them from unfetched
      candidates.
- Extract 2-4 abstract blueprints from the evidence rather
      than mirroring any single site.
- Prefer hybridized blueprint logic when multiple
      references fit closely.
- Keep the brief implementation-oriented, concise, and
      useful for art direction and planning.
- Make the selected blueprint explicit enough that
      downstream agents can follow its section rhythm,
      layout approach, and interaction pattern.
- Highlight interaction expectations, IA depth, primary
      user actions, and visual signals.
- Surface risks that could cause generic, off-genre, or
      drifted output.
- If source material is weak, still return a structured
      fallback brief instead of failing.
```
```

C.4 Aesthetic Specification Agent

The AESTHETIC SPECIFICATION AGENT converts the normalized request $\hat{R}$ and grounded brief G into the style contract S. Its role is to make abstract aesthetic intent operational before page planning, code generation, and asset generation. In the implementation schema, this contract is expanded into concrete fields such as palette, type, layout, motion, component grammar, imagery direction, and anti-patterns.

Style contract S:
"palette_system": {"background", "surface", "text", "accent", "contrast_pairs"},
"type_system": {"heading", "body", "scale", "hierarchy_rules"},
"layout_system": {"hero_style", "grid_style", "section_rhythm", "surface_treatment"},
"motion_system": {"mood", "entry_motion", "hover_motion", "ambient_animations"},
"component_grammar": {"cards", "buttons", "navigation", "media_treatment"}

Prompt.

Purpose
Define an implementation-ready visual system that maps the request and research context into palette, typography, layout, motion, and component grammar.

Inputs
- User request
- Research brief
- Experience mode
- Reference signals, design risk flags, curated references
- Implicit benchmark constraints

Output Contract
- `design_brief`
- `style_family`
- `palette_system`
- `type_system`
- `layout_system`
- `motion_system`
- `component_grammar`
- `anti_patterns`

Prompt Blocks
primary
```text
You are the art-direction agent in a multi-agent website generation pipeline.
Return STRICT JSON only.

Rules:
- Produce a distinctive, implementation-ready visual system.
- Bias toward strong design signals: typography hierarchy, layout rhythm, focal media, authored backgrounds, and controlled motion.
- Synthesize from the curated reference pool without copying literal branding or layouts.
- Keep the output useful to a frontend engineer; avoid vague mood-board language.
- Preserve implicit benchmark constraints as concrete art-direction obligations.
- Avoid generic dashboard, admin, and SaaS defaults unless the request clearly requires them.
- Do not default to purple-blue primaries or generic tech gradients unless the request or brand logic clearly supports them.
- Do not fall back to generic font stacks that erase brand character; choose heading/body pairings with visible contrast and purpose.
- Atmospheric gradients, glass layers, authored surfaces, and motion are allowed when they clarify hierarchy or brand tone; restraint does not mean visual timidity.
- Keep the visual system token-friendly: palette, surface, type, motion, and component decisions should read like a semantic design system, not a loose mood board.
- BACKGROUND DEFAULT: Unless the user request explicitly uses words like "dark", "dark mode", "night", "midnight", "black", "dark theme", or the brand clearly belongs to a dark-UI context (e.g. security tools, gaming, crypto, cinematic), set background and surface to light or neutral tones -- off-white, warm cream, light grey, or soft paper tones. Dark backgrounds are a deliberate branded choice, never a safe default.

LAYOUT SYSTEM RULES -- the layout_system field must be specific and varied:
- `hero_style`: choose a structurally distinct treatment -- options: "full-bleed split with oversized headline left, media right", "centered stage with editorial supertitle", "magazine-style diagonal crop with text overlay", "full-viewport media with anchored copy panel", "asymmetric hero with floating stat callouts". Never use "standard hero" or "banner".
- `section_rhythm`: mandate alternating density -- at least two distinct rhythm values, e.g. "alternating dense editorial and airy white-space sections; odd sections left-anchored, even sections reversed". Forbid "consistent layout throughout".
- `grid_style`: specify a concrete CSS strategy -- e.g. "12-column grid with editorial span-7 content and span-5 sidebar; feature cards use span-4 with one featured span-8"; NOT "grid layout" or "card grid".
- `surface_treatment`: mandate visual differentiation between sections -- e.g. "hero: full-bleed gradient; feature sections: tinted surface panels alternating left/right; stats: full-width contrast band; footer: recessed dark base". Each section type must have a named treatment.
- `background_treatment`: describe authored depth -- e.g. "radial light bloom on hero, subtle grain on feature sections, clean white for tool/form sections". Avoid "clean white background" as the only instruction.
- Layout anti-patterns to avoid: all sections same container width, all sections center-aligned, uniform card height, no full-bleed breaks, no typographic scale jumps between sections.

MOTION SYSTEM RULES -- the motion_system field must define a full per-section motion vocabulary:
- `mood`: one concrete phrase covering energy level and character, e.g. "confident and precise with moments of theatrical flourish" -- not just "smooth" or "minimal".
- `entry_motion`: assign a named entry preset to each section type from this vocabulary:
  - hero -> `clip_reveal` or `perspective_rise` (dramatic, above-the-fold)
  - feature/editorial sections -> `fade_up` with stagger OR `blur_in`
  - grid/card sections -> `stagger_children` (cards cascade in)
  - stats/numbers -> `fade_up` with staggered delays
  - media/gallery -> `media_zoom` on hover + `fade_up` on entry
  - footer/CTA -> `clip_reveal` or `fade_up`
  Never assign the same entry preset to every section.
- `hover_motion`: assign hover presets to interactive elements:
  - primary CTA buttons -> `magnetic`
  - feature/spotlight cards -> `tilt_3d` or `lift_glow`
  - media figures -> `media_zoom`
  - navigation links -> `underline_shift`
  - secondary cards -> `lift_glow`
  At least three distinct hover presets must appear across the page.
- `scroll_effects`: specify parallax strategy -- e.g. "hero background: subtle_parallax at 0.08 intensity; section dividers: clip-scroll; none on tool/form sections".
- `ambient_animations`: describe any always-on background animations -- e.g. "hero: slow-breathing gradient glow (opacity 0.6->0.8, 4s infinite); stats: number count-up on reveal; none elsewhere" -- or "none" if not appropriate.

CONTRAST PAIRING RULES -- every background variant in the palette_system must declare its paired text color, and the generator must not mix them arbitrarily:
- `background` (page default, light/neutral): text must be `text` (dark >= 4.5:1 against background); never use `muted` as the only text color on this surface.
- `surface` (card, panel, sidebar): text must be `text` or a dark variant; `muted` is allowed only for secondary/helper copy (never headings or primary body).
- `contrast_band` / full-bleed dark or saturated sections: text must be the inverse of the band color -- if band is dark (brightness < 128), text must be `#FFFFFF` or `background`; if band is light, text must be `text`.
- `accent` color: must never be used as body text or long-form copy color; use only for UI highlights, icons, and short labels where the surrounding background creates sufficient contrast.
- `gradient` / atmospheric backgrounds: every gradient surface must pair with a declared `on_gradient_text` value (light if the gradient is dark overall, dark if light); do not leave text color implicit on gradient surfaces.
- `glass` / transparent overlays: glass/frosted surfaces must set an explicit `color` on the overlay element so text does not inherit an unverified parent color; always add `text-shadow` or `backdrop-blur` to ensure legibility.
- The palette_system output must include a `contrast_pairs` array,
  This array is the authoritative pairing reference for the generator.
```

C.5 Plan Agent

The PLAN AGENT maps the style contract into page structure. It reads S, $\hat{R}$, and G, then produces the page outline O. Its role is to ensure that section sequencing, information flow, and interaction structure instantiate the contract rather than merely arranging content blocks.

Page outline O:
"page_goal", "section_sequence", "section_roles", "interaction_flow", "content_blocks", "contract_realization_notes".

Prompt.

```
## Purpose
Turn the request and art direction into a lightweight page
      contract that the generator can own and realize in
      browser-only HTML/CSS/JS.

## Inputs
- User request
- Research brief
- Design brief and locked visual system
- Implicit constraints and generation mode
- Optional leader directive

## Output Contract
- `page_outline`
- `experience_brief`
- `design_spec`
- `visual_direction`
- `inferred_features`
- `interaction_contract`

## Prompt Blocks
### outline
```text
You are a senior information architect planning a single-
      page website.
Return STRICT JSON only.

Rules:
- First section must be `site-nav`.
- Second section must be `hero`.
- Include at least six sections total: site-nav, hero, at
      least three body sections, and a footer-like close.
- `nav_anchor_targets` must cover every content section in
       scroll order.
- Preserve any required narrative order from constraints.
```

primary
```text
You are a senior product designer planning a single-page
      website.
Return STRICT JSON only.

Rules:
- Use the fixed outline exactly; do not rename or reorder
      sections.
- The locked art direction is authoritative.
- Prioritize narrative flow, focal media, typography
      hierarchy, and section rhythm before extra features.
- Keep only one or two core interactions unless the request
       clearly demands more.
- Required modules must appear as explicit modules, not
      dissolve into generic story copy.
- Explicit ecommerce, booking, dashboard, and tool
      functionality outrank inferred portfolio/editorial
      component bias.
- Interactive, booking, and ecommerce flows must become
      concrete browser interactions with testable DOM
      bindings.
- Pick a page archetype that fits the job: dashboard,
      workspace, wizard, feed, settings, or a narrative
      landing-page equivalent.
- For interaction-heavy or data-heavy pages, acceptance
      checks and interaction states should explicitly cover
       loading, empty, error, success, permission, and
      partial states.
- For narrative or mostly static pages, do not force all
      six states; keep only the states that map to real
      user interaction or gated content.
- Empty and error states should imply visible recovery
      paths, not just placeholder copy.
- Do not define final image counts, slot ids, placement
      coordinates, or drawing jobs.
- The generator owns layout, figure slots, DOM ids, and
      image placement.
- Your planning payload should stay light: page goal,
      section order, interaction contract, and acceptance
      checks.

LAYOUT VARIETY MANDATE -- every section in the page_outline
       must have a distinct `layout_mode` field chosen from
      :
  - `full_bleed_hero`: edge-to-edge media fill, copy
      anchored inside
  - `split_editorial`: two-column asymmetric split (e.g.
      55/45 or 60/40), alternating which side leads
  - `contained_editorial`: max-width container, strong
      typographic hierarchy, no card grid
  - `card_grid_staggered`: card grid with at least one
      featured/large card breaking the uniform pattern
  - `full_width_band`: 100vw contrast-color band, often
      used for stats, quotes, or CTAs
  - `magazine_mosaic`: overlapping or offset media/text
      blocks, editorial newspaper feel
  - `sticky_scroll`: sticky side panel with scrolling
      content alongside it
  - `centered_stage`: centered composition, cinematic
      framing, used for primary CTA or manifesto sections
  - `tool_workspace`: form, filter, or interactive tool
      with result area, functional layout
- No two adjacent sections may share the same layout_mode.
- The hero must use `full_bleed_hero`, `split_editorial`,
      or `centered_stage`.
- At least one section must use `full_width_band` (for a
      stat strip or quote callout).
- At least one section must use `card_grid_staggered` or `
      magazine_mosaic` for content variety.

MOTION ASSIGNMENT MANDATE -- every section must have a `
      motion_preset` field:
  - hero -> `clip_reveal` or `perspective_rise`; trigger:
      `load`
  - editorial/feature sections -> `fade_up` or `blur_in`;
      trigger: `in_view`
  - card/grid sections -> `stagger_children`; trigger: `
      in_view`
  - stat/counter sections -> `fade_up` with per-child
      stagger delays; trigger: `in_view`
  - CTA/footer sections -> `clip_reveal` or `fade_up`;
      trigger: `in_view`
- No two adjacent sections may share both the same
      motion_preset AND the same trigger.
- Primary CTA buttons must have hover_preset: `magnetic`.
- Spotlight/story cards must have hover_preset: `tilt_3d`.
- Feature cards must have hover_preset: `lift_glow`.
- Media figures must have hover_preset: `media_zoom`.

SECTION RHYTHM MANDATE -- the design_spec must describe
      section differentiation:
- Alternate between "dense" (information-rich, tighter
      spacing) and "airy" (open, typographic focus)
      sections.
- At least one section must use a full-bleed background
      that differs from the page background.
- At least one section must have a typographic scale jump (
      display/oversized heading >= 5rem or stat numbers >=
      6rem).
- Forbid same padding, same max-width, and same text
      alignment across all sections.
```

repair
```text
You are repairing only the planning payload.
Return STRICT JSON only with the same outer shape requested
       by the caller.

Rules:
```
```

```
- Do not introduce illustration tasks, slot ids, placement
      coordinates, or HTML fragments.
- Keep the outline, experience brief, design spec, inferred
      features, and interaction contract aligned.
- Preserve a clear hero section and a realistic interaction
      contract.
```

C.6 Generator Agent

The GENERATOR AGENT reads the style contract S and page outline O, then produces the executable front-end implementation C and auxiliary illustration tasks T_I. Its role is to realize the contract in browser-executable code while preserving the intended structure, component behavior, and asset slots.

Implementation bundle (C, T_I)**:**
"html", "css", "javascript", "asset_manifest", "illustration_tasks", "integration_notes".

Prompt.

```
## Purpose
Generate a complete front-end bundle document that can be
      extracted into local HTML, CSS, and JS while
      preserving structure, interaction contracts, the
      visual system, and generator-owned image slots.

## Inputs
- Page outline
- Experience brief
- Design spec and visual direction
- Art-direction systems and component recipes
- Interaction contract
- Generator-owned asset manifest on integration pass
- Illustration results on integration pass
- Generation profile

## Output Contract
- Scaffold pass: HTML only, with one inline style block and
       one inline script block ready for extraction
- Integration pass: final HTML with resolved illustration
      assets merged into existing slots

## Prompt Blocks
### primary
```text
You are a senior frontend engineer creating a complete
      website entry document that will be externalized into
       a local asset bundle.

Rules:
- Output only HTML.
- Include `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`,
       `<main>`, and `<nav>`.
- Emit one inline stylesheet block and one inline script
      block for later extraction.
- Implement every section id from the outline.
- Use only local/self-contained assets; no remote CSS,
      fonts, images, or scripts.
- Treat visual direction, component composition, and motion
       contracts as hard implementation constraints.
- Real user-facing interactions must be implemented with
      browser-visible DOM and JS behavior, not just visual
      suggestion.
- If generation mode is interactive, include a focused
      playable loop with visible controls, state, and reset
      .
- Scaffold pass owns all figure slots, `data-asset-id` / `
      data-slot-id` metadata, and slot layout behavior.
- Illustration figures must remain supportive visual slots
      rather than carry core UI or data.
- Integration pass should adapt layout, overlay, and
      readability after real images arrive instead of
      asking planning stages to change the page structure.
- Give the page at least one unmistakable visual anchor,
      especially in the hero or primary interaction zone.
- Make interaction feedback perceptible: hover, press,
      loading, success, and error states should be visible
      without close inspection.
- Push at least one core component beyond bare utility into
       an authored, design-led treatment rather than
      shipping only functionally correct boxes.
- Honour the palette_system background colour exactly. Do
      NOT override a light background with a dark one
      unless visual_direction explicitly specifies a dark
      theme.

LAYOUT TECHNIQUES -- use concrete CSS to create variety
      across sections:
- Hero: use CSS Grid with `grid-template-areas` to anchor
      copy and media independently; allow text to overlap
      the image with a z-index layer and semi-transparent
      scrim if needed.
- Split editorial sections: use `display: grid; grid-
      template-columns: 55fr 45fr` (or reversed); alternate
       left/right lead on consecutive sections.
- Full-width bands (stats, quotes, CTA): use `width: 100vw;
       margin-inline: calc(50% - 50vw)` to break out of the
       container, with a distinct background-color.
- Card grids: default to `grid-template-columns: repeat(
      auto-fill, minmax(280px, 1fr))` with stable single-
      span cards. Create rhythm through content contrast,
      media ratio, and section pacing rather than `grid-
      column: span 2` or `grid-row: span 2` in ordinary
      card collections.
- Default card collections to a stable variable-height grid
      , not freeform masonry. Stable grids should avoid
      featured spans and sparse tail rows; only explicit
      editorial/feature layouts may opt into asymmetric
      multi-span cards.
- Magazine/mosaic layouts: use `position: relative` with `
      top`/`left` offsets or negative margins on
      alternating child elements to create editorial
      overlap.
- Sticky sidebars: use `position: sticky; top: 2rem` on a
      side column alongside a scrolling content column.
- Typography scale jumps: use `font-size: clamp(3.5rem, 8vw
      , 7rem)` for display headings, `font-size: clamp(0.6
      rem, 1vw, 0.75rem)` for eyebrow labels. Every section
       should have a different typographic scale role.
- Section padding variation: alternate between `padding-
      block: 6rem 8rem` (airy) and `padding-block: 3rem 4
      rem` (dense). Do not use the same padding on every
      section.
- Use CSS custom properties (`--section-bg`, `--section-
      text`) per section so each section can override
      surface without global style conflicts.

MOTION IMPLEMENTATION -- annotate elements precisely:
- Every `<section>` must have `data-motion="true"` and `
      data-motion-trigger` set to `"load"` (hero/above-fold
      ) or `"in_view"` (all others).
- **CRITICAL -- `data-motion-trigger="active_state"` is
      for interactive result sections ONLY.** A section
      should use `active_state` only if its `section_id`
      appears in the motion_contract's `
      interactive_component_targets` list AND the section
      displays content that changes when the user submits a
       form (search results, availability panels, etc.).
      Gallery sections, editorial sections, and product
      grids that are always visible MUST use `"in_view"`,
      not `"active_state"`. Using `"active_state"` on a
      static section means it will never reveal because the
       MutationObserver only fires on DOM changes, not on
      scroll.
- `story_spotlight_panel` is the one allowed non-form
      exception: it must bind `active_state` through a
      custom `activeStateChanged` event, ship a default
      visible state, and never rely on stock `result_ids`
      mutation wiring.
- Use `data-motion-entry` to pick the entry preset that
      matches the section's content type:
  - Full-bleed hero, CTA stage -> `clip_reveal`
  - 3D card or spotlight panel -> `perspective_rise`
  - Image-heavy or atmospheric section -> `blur_in`
  - Bold headline/manifesto section -> `clip_reveal`
  - Card grid or feature list -> `stagger_children`
  - Default for everything else -> `fade_up`
```
```

```
- For `stagger_children` sections: add `data-motion-child
      ="true"` and `data-motion-delay="Nms"` (0, 80, 160,
      240...) on each child card directly in HTML so they
      cascade in independently.
- Use `data-motion-hover` on interactive elements:
  - Primary CTA buttons -> `magnetic`
  - Spotlight / story cards -> `tilt_3d`
  - Feature cards, stat panels -> `lift_glow`
  - Media figures -> `media_zoom`
  - Text navigation links -> `underline_shift`
- Add `data-motion-delay` to hero child elements (headline
      , subtitle, CTA) with incremental offsets (0ms, 120ms
      , 240ms) to create a natural read-order cascade.
- For stat/counter sections: add a `data-count-to`
      attribute on numeric spans and implement a JS count-
      up animation triggered by IntersectionObserver.
- Ambient CSS animations (infinite breathing glows, slow
      gradient drifts, floating particles) are encouraged
      for hero and atmospheric sections -- use `animation-
      iteration-count: infinite` freely for these non-
      scroll effects.
- Always include `@media (prefers-reduced-motion: reduce)`
       rules that disable all transitions, animations, and
      transforms.

VISUAL RHYTHM CHECKLIST -- before finalising the output,
      verify:
1. No two adjacent sections share the same background color
       AND the same max-width container.
2. At least one section breaks to full viewport width (100
      vw).
3. At least one card or feature item is visually larger
      than its siblings (featured/hero card).
4. The hero has a dominant above-the-fold visual anchor (
      full-bleed figure, authored gradient, or both).
5. At least three distinct `data-motion-entry` presets are
       used across the page.
6. At least three distinct `data-motion-hover` presets are
       used across interactive elements.

ILLUSTRATION SLOT RULES -- static figures are mandatory for
       all content images:
- Every image that represents real content (room photo,
      product shot, portfolio piece, scene, environment,
      person) MUST be a static `<figure class="illustration
      " data-asset-id="UNIQUE-ID" data-illustration-id="
      UNIQUE-ID" ...><!--ASSET:UNIQUE-ID--></figure>`
      element written directly in the HTML. The
      illustration pipeline can only inject images into
      these static elements.
- FORBIDDEN pattern -- JS data arrays with image paths:
  ```js
  const rooms = [{ img: 'index_assets/img/room-1.jpg', name
      : '...' }];  // [BAD] NEVER DO THIS
  ```
  These paths are invented, will never resolve, and bypass
      the illustration injection system entirely.
- ALLOWED pattern -- static figures that JS can filter/sort
      /show/hide:
  ```html
  <figure class="illustration" data-asset-id="room-suite"
      data-illustration-id="room-suite"
          data-filter-type="suite" data-filter-guests
      ="4"><!--ASSET:room-suite--></figure>
  <figure class="illustration" data-asset-id="room-deluxe"
      data-illustration-id="room-deluxe"
          data-filter-type="deluxe" data-filter-guests
      ="2"><!--ASSET:room-deluxe--></figure>
  ```
  JS reads `data-filter-*` attributes to show/hide cards
      without ever generating `<img>` tags itself.
- If a section needs a JS-driven gallery, filterable grid,
      or room finder: write all cards as static HTML first
      (with figure slots), then attach JS filtering/sorting
       behaviour on top using `data-*` attributes. Never
      use `innerHTML` or `createElement` to inject
      illustration images.
- Exception: purely decorative CSS background-image values
      driven by JS are allowed if they reference a known `
      data-asset-id` path that was already injected by the
      illustration pipeline.

INTERACTION RESULT DISPLAY RULES -- search/availability
      results must clone existing static cards:
- When an interaction (availability check, search, filter,
      recommendation) needs to display results that
      correspond to items already represented as static
      cards in the page, the JS MUST clone those existing
      DOM nodes -- never create new card HTML via `
      innerHTML` or `createElement`.
- FORBIDDEN pattern -- recreating cards in the result
      container:
  ```js
  resultsContainer.innerHTML = `
    <div class="room-card">           <!-- [BAD] no image,
      wrong structure -->
      <div class="room-card__content">
        <h3>Valley Deluxe</h3>
      </div>
    </div>`;
  ```
  This loses the image slot entirely and produces visually
      inconsistent cards.
- REQUIRED pattern -- clone existing static cards and
      augment with result data:
  ```js
  const source = document.querySelector('#stay .room-card[
      data-room-type="deluxe"]');
  if (source) {
    const clone = source.cloneNode(true);           // deep
       clone preserves <figure> + <img>
    clone.removeAttribute('data-motion-child');       //
      strip stagger so card appears immediately
    clone.removeAttribute('data-motion-delay');
    const content = clone.querySelector('.room-
      card__content');
    if (content) {
      // append price/date info; update CTA text/class
      const cta = content.querySelector('a.btn');
      if (cta) { cta.textContent = 'Book Now'; cta.
      className = 'btn btn--primary'; }
    }
    resultsContainer.appendChild(clone);
  }
  ```
  The cloned card keeps its `<figure class="illustration
      ">` with the already-loaded `<img>`, guaranteeing
      visual and structural consistency with the catalogue
      cards.
- If a result item has NO corresponding static card (e.g.,
      the search returns a new item not in the catalogue),
      it is acceptable to create a card via `createElement
      `, but it MUST include a `<div class="[component]
      __media">` wrapper and a `<figure class="illustration
      ">` slot -- even if the figure shows a placeholder
      colour. Never omit the media section entirely.
- Do NOT append supplemental `<figure class="illustration
      ">` nodes directly to the end of a `<section>`. Real
      content images must live inside a card media wrapper,
       an existing collection container, or an explicit
      illustration slot rail.
- FORBIDDEN: Using `<!--ASSET:${id}-->` or `<!--
      ILLUSTRATION:${id}-->` comment placeholders inside JS
       template literals -- these comments are the pipeline
      's pre-injection markers and are only resolved during
       the static HTML processing pass. Dynamically created
       figures with these comments will never receive their
       images at runtime.
  ```js
  // [BAD] NEVER: pipeline placeholder inside a template
      literal
  const html = `<figure data-illustration-id="${room.
      assetId}"><!--ASSET:${room.assetId}--></figure>`;
  // [OK] CORRECT: clone the already-resolved static figure
  const src = document.querySelector('.illustration[data-
      illustration-id="' + room.assetId + '"]');
  const figClone = src ? src.cloneNode(true) : null;
  ```

HERO ILLUSTRATION RULES -- hero images must always use a
      landscape aspect ratio:
- The hero's main illustration figure MUST declare a
      landscape `data-target-aspect`: use `"16/9"`,
      `"3/2"`, or `"2/1"`. NEVER use `"9/16"`, `"2/3"`, or
       any portrait ratio for the hero.
- Portrait aspect ratios on hero images produce a narrow
      (~67vw wide) image that leaves a large empty gap on
      the right side of the hero, breaking the full-bleed
      visual anchor.
- If the hero section needs to show a portrait-oriented
      photo (e.g. a person), crop it to landscape in the
      prompt and use `data-framing="wide shot"`.
```

```
- FORBIDDEN: `<figure ... data-target-aspect="9 / 16" ...>` in the hero section
- REQUIRED: `<figure ... data-target-aspect="16 / 9" ...>` (or "3/2" or "2/1") in the hero section

ILLUSTRATION SIZING RULES -- figure containers must be sized with aspect-ratio, not fixed heights:
- Define illustration container dimensions using `aspect-ratio` CSS only (e.g., `style="aspect-ratio: 16/9;"`). NEVER use `height: Npx` or `height: Nrem` on a `<figure class="illustration">` element.
- Fixed heights conflict with the post-generation aspect-ratio enforcement pass: if the figure has `height: 400px`, the injected `aspect-ratio` is silently ignored by the browser, causing the image to fill an arbitrary non-matching shape.
- For card grids where cards have a uniform visual height, set `min-height` on the CARD wrapper, not on the inner `<figure>`. Let the `<figure>` define its own ratio inside the card.
- Card collections may use variable content height, but the grid itself must stay top-aligned: use `align-items: start`, `align-content: start`, and `grid-auto-rows: auto` on the collection container.
- The image generation API snaps all requested sizes to one of three outputs: 1024x1024 (square), 1536x1024 (landscape 3:2), 1024x1536 (portrait 2:3). The post-processing pipeline will update your declared `aspect-ratio` to match the actual generated size -- so your initial value is a hint, not a guarantee. Design your layout to tolerate a +/-20% ratio shift.
- `object-fit: cover` is the default for all illustration images: slight edge cropping is acceptable; letterbox bars are not.

CONTRAST RULES -- legibility is a hard constraint, not a design option:
- Whenever you override `--section-bg` to a dark color (brightness < 128), you MUST also set `--section-text` to a light value (`#FFFFFF` or `var(--palette-background)`) on the same element or its `:root` scope for that section.
- Whenever you override `--section-bg` to a light/neutral color, you MUST set `--section-text` to a dark value (`var(--palette-text)` or equivalent).
- DARK SECTION BUTTON RULE -- `.btn--secondary` uses the global `--palette-text` (dark) by default. Inside any dark-background section you MUST add a scoped override: `#section-id .btn--secondary { color: var(--section-text); border-color: rgba(255,255,255,0.3); }`. Without this, secondary buttons are invisible (dark text on dark background). This applies to ALL secondary buttons inside dark sections: navigation buttons, "Back" buttons in multi-step forms, "Request Details" links, etc.
- Do NOT use `--palette-muted` (typically a low-contrast grey) as the only text color on a colored or dark section background -- reserve muted for secondary copy on the default page background only.
- Gradient backgrounds: every gradient section must add a semi-transparent scrim (`background: linear-gradient(to bottom, rgba(0,0,0,0.45), rgba(0,0,0,0.45))` overlay or equivalent) behind text, OR use light text with `text-shadow: 0 1px 4px rgba(0,0,0,0.6)`.
- Glass / frosted overlays: always set an explicit `color` on the glass element; never rely on inherited color that may clash with the blurred background.
- `--palette-accent` must NOT be used as body text, paragraph copy, or long-form text color -- use it only for UI highlights (borders, icons, labels <= 4 words, button text on accent-background).
- Full-width contrast bands (`.full-width-band`, stat strips, quote callouts): if the band uses a dark background, ALL child text elements (`h2`, `p`, `span`, `strong`) must inherit a light text color explicitly set on the band container, not just through `color: inherit`.
- After setting a section background, always verify the text color is set explicitly on the same element or a direct child wrapper -- do not rely solely on CSS variable inheritance across arbitrary DOM depth.

NAV STRUCTURE RULES -- every navigation link must live inside a list item:
- ALL `<a>` tags inside `<nav>` MUST be wrapped in `<ul><li><a>` -- never place a bare `<a>` directly as a child of `<nav>` or after the closing `</ul>`.
- FORBIDDEN pattern:
  ```html
  <nav>
    <ul>
      <li><a href="#hero">Home</a></li>
    </ul>
    <a href="#contact">Contact</a>  <!-- [BAD] bare anchor outside <ul> -->
  </nav>
  ```
- REQUIRED pattern:
  ```html
  <nav>
    <ul>
      <li><a href="#hero">Home</a></li>
      <li><a href="#contact">Contact</a></li>  <!-- [OK] every link in <li> -->
    </ul>
  </nav>
  ```
- Navigation links for hamburger/mobile menus, social icons, and CTAs that sit outside the main `<ul>` are allowed ONLY if they are inside their own separate `<ul>` -- never as floating `<a>` elements.

FORM INPUT RULES -- every visible input must have an accessible label:
- Every `<input>` that the user can interact with (type text, email, tel, number, date, search, password, url, textarea) MUST have either:
  - a `<label for="INPUT_ID">` that matches the input's `id`, OR
  - an `aria-label="..."` attribute on the input itself.
- FORBIDDEN pattern:
  ```html
  <input type="text" id="check-in" placeholder="Check-in date" />  <!-- [BAD] no label -->
  ```
- REQUIRED pattern:
  ```html
  <label for="check-in">Check-in date</label>
  <input type="text" id="check-in" placeholder="Check-in date" />  <!-- [OK] paired label -->
  ```
- If a visible label would break the design, use `aria-label` instead -- but never omit both. Unlabeled inputs will be auto-hidden to `type="hidden"` by the post-processing pipeline.

SECTION SPACING RULES -- vary rhythm, avoid uniform padding across all sections:
- Hero section padding is unconstrained (commonly `min-height: 100vh`).
- For all other sections, use ONE of three padding tiers:
  - Dense: `padding-block: 3rem 4rem` -- for stat strips, quote bands, narrow CTA bars
  - Standard: `padding-block: 5rem 6rem` -- for editorial splits, feature grids, testimonials
  - Airy: `padding-block: 7rem 8rem` -- reserved for the FIRST non-hero content section only
- NO more than one consecutive non-hero section may use 7rem+ padding. If two adjacent sections both want "airy", give the first one standard and the second airy.
- The closing CTA / booking section should always use dense or standard -- NOT airy -- to avoid a cavernous gap before the footer.
- Apply these as section-specific CSS rules (e.g. `#amenities { padding-block: 5rem 6rem; }`) not as a single global value repeated on every section.

CARD MEDIA WRAPPER RULES -- card image containers must clip content:
- Every card media wrapper (the direct parent of an illustration figure inside a card) MUST have `overflow: hidden` in its CSS rule. This prevents figures with `aspect-ratio` from bleeding into adjacent grid columns.
- FORBIDDEN: `<div class="room-card__media">` without `overflow: hidden`
- REQUIRED: `.room-card__media { overflow: hidden; }` (or equivalent for your class name)
- Every real card must keep a stable two-part DOM: `card root -> media wrapper + content wrapper`. Do not place the `<figure class="illustration">` as a loose
```

```
      sibling after the card body or at section tail.
- NEVER set `height: 100%` on a `<figure class="
      illustration">` inline style. Use `height: 100%` on
      the WRAPPER div instead, and let the figure use only
      `aspect-ratio`. This prevents the browser from
      computing an oversized width from the height + aspect
      -ratio combination.

FONT FALLBACK RULES -- always declare system fallbacks for
      every font stack:
- Every `font-family` declaration MUST include at least
      one generic system fallback (`sans-serif`, `serif`, `
      monospace`).
- FORBIDDEN: `font-family: 'Lora', 'Inter';` (no fallback)
- REQUIRED: `font-family: 'Lora', Georgia, 'Times New Roman
      ', serif;`
- Since external fonts are forbidden (no remote `@import`
      or CDN `<link>`), all named fonts must be available
      as system fonts on macOS/Windows or declared via
      inline `@font-face`. When in doubt, omit the custom
      name and use the system stack directly.
```

C.7 Illustration Agent

The ILLUSTRATION AGENT realizes visual assets V from illustration tasks T_I under the style contract S. Its role is to prevent imagery from drifting away from the page-level design direction.

Asset set V:
"assets": [{"id", "section_id", "type", "semantic_role", "style_notes", "path"}],
"decorative_elements": [{"id", "usage", "palette_binding", "placement"}],
"integration_notes": {"crop_or_fit", "contrast_safety", "section_alignment"}

Prompt.

```
## Purpose
Generate or fallback-render visual assets that support the
      page without blocking HTML delivery.

## Inputs
- Drawing job generated from the generator-owned asset
      manifest
- Design brief / design spec
- Optional leader directive

## Output Contract
- `illustration_results`
- Each drawing job resolves to raster output, SVG output,
      or deterministic fallback SVG

## Prompt Blocks
### raster
```text
Create a pure visual asset for a generated website.

Rules:
- This is not a UI mockup, infographic, dashboard, chart,
      or application screen.
- Do not include readable text, labels, numerals, browser
      chrome, or controls.
- Favor one focused subject, prop, scene, icon, texture, or
       background moment.
- Keep the asset semantically light so the real UI remains
      in HTML/CSS/JS.
- Respect slot-aware framing, aspect ratio, and subject
      constraints from the drawing job.
- Preserve believable color separation and avoid muddy gray
       haze unless the brief explicitly requests monochrome
      .
```

svg
```text
Create one self-contained SVG illustration for a website.

Rules:
- Return only `<svg>...</svg>`.
- Match the requested viewBox ratio when possible.
- Include `<title>` and `<desc>`.
- No scripts, external images, charts, text labels, data
      panels, or browser UI motifs.
- Treat the asset as supportive visual atmosphere rather
      than a functional interface.
```
```

C.8 Validation Agent

The VALIDATION AGENT checks whether the generated implementation C, assets V, and rendered webpage remain faithful to the normalized request $\hat{R}$, grounded brief G, page outline O, and style contract S. It produces the validation report $Q = (E, \Omega, \Delta)$, where E records blocking errors, Ω records warnings, and Δ contains candidate revision directives.

Validation report Q:
"errors": [{"type", "severity", "description", "source"}],
"warnings": [{"type", "description", "affected_scope", "priority"}],
"suggested_actions": [{"target", "operation", "rationale"}]

Prompt.

```
## Purpose
Validate the generated front-end bundle and run an isolated
       rendered review when available without sacrificing
      final HTML delivery.

## Inputs
- Generated HTML / bundle
- Outline, interaction, motion, and illustration
      specifications
- Optional rendered screenshots for vision review

## Output Contract
- Validation report with errors, warnings, and suggested
      follow-up actions

## Prompt Blocks
### vision_review
```text
You are reviewing rendered screenshots of a single-page
      website.
Return only valid JSON with shape:
{
  "issues": [
    {
      "severity": "warn",
      "summary": "short finding"
    }
  ]
}

Rules:
- Base findings only on what is visible in screenshots.
- Use severity `warn` for every item.
- If the page looks acceptable, return `{"issues": []}`.
- Focus on overlap, contrast, broken imagery, readability,
      and layout polish.
```
```

C.9 Requirement-Aware Arbitration

After the VALIDATION AGENT produces Q, the USER AGENT performs requirement-aware arbitration. It compares the validation report with the normalized request $\hat{R}$ and the style contract S, then decides whether the webpage should be accepted or locally repaired:

$$(y, \rho) = A^{\text{arb}}_{\text{user}}(\hat{R}, S, Q).$$

Here, $y \in \{0, 1\}$ is the acceptance decision and ρ contains localized repair instructions. If $y = 1$, the webpage is accepted. If $y = 0$, the instructions in ρ are routed to the relevant downstream agents for targeted repair.

D WebJudge Evaluation Protocol

This section provides the full WebJudge evaluation protocol, including requirement separation, rendered visual evidence, GUI-agent testing, functional and content checklists, prompt templates, pairwise comparison, and item-level visual-aesthetic scoring anchors. WebJudge is a structured multimodal evaluator: it separates visual-style requirements from non-visual requirements, evaluates rendered webpage evidence with a fixed VisAWI-Pos-inspired rubric, and evaluates content and function through checklist and GUI-agent evidence.

D.1 Requirement Separation

Before scoring, WebJudge separates each brief into visual-style requirements and non-visual requirements. Visual-style requirements are evaluated through the rendered appearance of the webpage. Content and functional requirements are converted into checklist items and GUI-agent testing instructions. This separation prevents stylistic requests from being treated as ordinary functional failures, while also preventing visually polished pages from receiving high scores when required content or interactions are missing.

The requirement-separation output contains:

- **Visual-style requirements**: requirements about the intended visual identity, mood, layout feel, interaction feel, imagery direction, and overall design atmosphere.
- **Functional requirements**: requirements about executable interactions, navigation, forms, filters, shopping flows, booking actions, or other user actions.
- **Content requirements**: requirements about required sections, messages, information hierarchy, target audience, offerings, and communicative goals.
- **Ambiguous or excluded requirements**: requirements that cannot be reliably assessed from the rendered page or are outside the front-end scope.

D.2 Rendering and Evidence Collection

Each generated webpage is served locally and rendered in a controlled Chromium environment. WebJudge captures a full-page screenshot and uses it as visual evidence for the aesthetic rubric. The visual evaluator judges the rendered webpage, not the source code or intermediate agent artifacts. For functional and content evidence, WebJudge records GUI-agent trajectories, visible headings and text, available interface elements, URLs, and executed actions.

D.3 Visual-Aesthetics Rubric

For the visual component, a multimodal evaluator applies a fixed VisAWI-Pos-inspired rubric to the rendered page. Table 9 summarizes the four visual dimensions, while Tables 10–13 provide item-level scoring anchors. These anchors make the multimodal evaluation structured and reproducible, avoiding an unconstrained holistic preference prompt.

Table 9: VisAWI-Pos-inspired visual rubric used by WebJudge.

Dimension	Item-level criteria
Simplicity	Layout cleanliness, comprehensibility, structure, uniformity, and coherence.
Diversity	Visual interest, inventiveness, inspiration, dynamism, and variation.
Colourfulness	Attractiveness, appropriateness, matching, and appeal of the colour composition.
Craftsmanship	Professional design quality, modernity, careful construction, and conceptual clarity.

D.4 Functional and Content Checklists

For functional validity, WebJudge converts functional requirements into GUI-agent testing instructions. The GUI agent interacts with the locally served webpage, records executed actions and visible evidence, and summarizes whether each requirement is satisfied. For content alignment, WebJudge checks whether the non-visual informa-

tion, section structure, and required communicative content from the brief are present. Purely stylistic requirements are excluded from these checklists and evaluated by the visual-aesthetics rubric.

D.5 Score Aggregation and Pairwise Comparison

WebJudge reports separate scores for visual aesthetics, functional validity, and content alignment. The visual score is computed from the VisAWI-Pos-inspired dimensions. Functional and content scores are computed from requirement-level checklist outcomes. The overall score used in the main experiments combines the three task dimensions after normalization. Pairwise win rates are computed by comparing methods on the same Web4Style brief under the same evaluation protocol.

D.6 Prompt Templates

The following prompts instantiate the main stages of WebJudge. The prompt boxes are preserved verbatim for reproducibility.

GUI-Agent Test Instruction Prompt.

```
function_test_prompt = """Based on the original website
      development instruction, identify the testable
      requirements and create one instruction for a web-
      navigation GUI agent to test the generated website.
      The following is an example of how to derive a GUI-
      agent testing instruction from the original
      instruction:

<example>
Original instruction: Please implement a museum website for
       a contemporary art center. The website should allow
      users to browse current exhibitions, view event
      schedules, learn about artists, and book tickets for
      selected events. Users should also be able to search
      exhibitions and check visitor information such as
      opening hours and location.

<boltAction type="gui_agent_test">Browse current
      exhibitions, inspect artist and event information,
      check visitor information, search for exhibitions,
      and verify that ticket-booking-related interactions
      can be completed for selected events.</boltAction>
</example>

The following is the original website development
      instruction:

<instruction>{instruction}</instruction>

Write one concise GUI-agent testing instruction in the same
       format as the example. Do not output any explanation
      .

Important constraints:
- Focus on requirements that can be verified from rendered
      content and interaction.
- Prioritize meaningful user flows over cosmetic checks.
- Do not include purely visual requirements such as color,
      typography, or style.
"""
```

Functional and Content Evaluation Prompt.

```
Functional_Content_PROMPT = """You are evaluating a GUI-
      agent testing trajectory for a generated website.

Original instruction:
{instruction}

GUI-agent testing instruction:
{gui_instruction}

GUI-agent final verdict:
{agent_verdict}

Structured observations:
{structured_observations}

Site-wide content evidence:
{site_content_evidence}

Action trace summary:
{trajectory}

Return valid JSON only:
{{
  "summary": "<string>",
  "test_passed": <boolean>,
  "improvement_suggestions": "<string>",
  "dimension_results": {{
    "functional_validity": <int 0-100>,
    "content_alignment": <int 0-100>
  }},
  "coverage_details": {{
    "functional": {{
      "covered_count": <int>,
      "total_count": <int>
    }},
    "content": {{
      "covered_count": <int>,
      "total_count": <int>
    }}
  }},
  "requirement_checks": {{
    "functional": [
      {{
        "requirement": "<string>",
        "covered": <boolean>,
        "evidence": "<string>"
      }}
    ],
    "content": [
      {{
        "requirement": "<string>",
        "covered": <boolean>,
        "evidence": "<string>"
      }}
    ]
  }},
  "dimension_summaries": {{
    "functional_validity": "<string>",
    "content_alignment": "<string>"
  }}
}}

Rules:
- `summary` should concisely state what worked and what
      failed.
- Evaluate with a practical product-review mindset, but use
       the structured evidence strictly.
- Split requirements into two groups only:
  - `functional`: interaction requirements that must be
      verified through actual user actions and resulting
      page changes.
  - `content`: non-visual semantic content requirements
      verifiable from rendered page text, headings,
      navigation, sections, cards, FAQ, contact info,
      product categories, or similar visible structure.
- Exclude purely visual requirements such as color,
      typography, atmosphere, layout beauty, or stylistic
      tone.
- For `functional` requirements, only mark `covered: true`
       when the interaction was actually verified by
      evidence from actions, URLs, page transitions,
      changed state, or other explicit structured
      observations.
```

```
- For `content` requirements, use the site-wide content
      evidence as the primary source of truth. Mark `
      covered: true` when the content appears in rendered
      evidence anywhere in the sampled site pages. Do not
      count content that only exists in source code,
      comments, hidden templates, or unvisited dead routes.
- Content requirements may be satisfied by any sampled page
       on the site, not only the exact page the GUI agent
      visited during interaction testing.
- Accept semantic equivalents for content and navigation
      labels when the same user intent is clearly satisfied
      .
- `test_passed` is only an auxiliary summary field. It can
       be true when most core functional requirements are
      covered, even if some secondary issues remain.
- `dimension_results.functional_validity` must equal `
      round(functional.covered_count / functional.
      total_count * 100)`, or `0` when `functional.
      total_count == 0`.
- `dimension_results.content_alignment` must equal `round(
      content.covered_count / content.total_count * 100)`,
      or `0` when `content.total_count == 0`.
- `coverage_details` counts must be non-negative integers
      with `covered_count <= total_count`.
- `requirement_checks.functional` must contain every
      requirement counted in `coverage_details.functional.
      total_count`, no extras.
- `requirement_checks.content` must contain every
      requirement counted in `coverage_details.content.
      total_count`, no extras.
- `dimension_summaries` must briefly explain the evidence
      for each dimension.
"""
```

Style Evaluation Prompt.

```
STYLE_PROMPT = """You are evaluating a generated website
      against the original build instruction. You are given
       one full-page screenshot of the page.

Inputs:
Original instruction: {instruction}
Rubric: {rubric}
Scoring anchors: {anchors}

Task:
1. Check whether the screenshot shows a rendering or
      runtime error.
2. If there is no error, describe the page style across the
       full page using this screenshot.
3. Suggest concrete style improvements only when necessary.
4. Extract MUST-HAVE VISUAL requirements from the
      instruction that are verifiable via screenshot, then
      check whether each is present.
5. Apply the hard caps from "Visual-brief alignment rules"
      only when visible style requirements are missing.
6. Score every rubric item from 1 to 7 based on the overall
       page impression, after applying caps.

Return valid JSON only. The response must contain `error`,
       `description`, `suggestions`, `
      must_have_requirements`, and `scores`. Each element
      in `must_have_requirements` contains `requirement`, `
      present`, and `evidence`. The `scores` field follows
      {scores_schema}.

Rules:
- Use `error` for visible rendering/runtime failures;
      otherwise use an empty string.
- Keep `description` focused on the rendered UI across the
       full page.
- If no improvement is needed, `suggestions` should be an
      empty string.
- The `must_have_requirements` list must have 3--8 VISUAL
      requirements, each verifiable by screenshot.
- Every rubric item must be present exactly once in `scores
      `; every score must be an integer from 1 to 7.
- Be strict about visible style alignment only: if key
      requested visual elements are missing, do not award
      top scores in the affected items.
- A page can be attractive yet still score only 4--5 if it
      feels generic, repetitive, incomplete, weakly aligned
       to the brief, or lacking exceptional polish.
- Narrative flow, section rhythm, and concept expression
      should influence existing simplicity, diversity, and
      craftsmanship items even when all required modules
      are present.
- Ignore missing business modules, exact navigation labels,
       feature completeness, or information-architecture
      completeness unless they directly create visible
      placeholders, broken rhythm, or obvious layout
      degradation in the screenshot.
- If visible style treatments are present but expressed
      weakly, generically, or with repetitive treatment,
      keep the affected scores in the 4--5 range.
- Do not treat mere neatness, conservative structure, or
      wireframe-like polish as enough for top scores when
      another page could express the brief more richly.
- Reserve score 7 for truly exceptional, rare, and tightly
      brief-aligned cases only. Most strong pages should
      top out at 5--6.
"""
```

Visual Pairwise Comparison Prompt.

```
VISUAL_COMPARE_PROMPT = """You are comparing two website
      screenshots side by side.

Left screenshot: website A
Right screenshot: website B

Task:
1. Compare the two full-page screenshots directly.
2. For each rubric item, decide which side is better: `left
      ` or `right`.
3. For each visual dimension, decide which side wins
      overall.
4. Decide one overall visual winner.
5. Provide a concise rationale.

Important rules:
- Do NOT independently score each site first and then
      compare.
- Directly judge which side is better for each item and
      dimension.
- Judge only the visible visual result in the screenshots.
- Ignore business-module completeness, exact navigation
      labels, information architecture completeness, and
      feature omissions unless they create obvious visible
      placeholders, broken rhythm, or clear layout
      degradation in the screenshot.
- When both pages look attractive, prefer the side with
      stronger narrative flow, section rhythm, mood fit,
      concept completion, richer imagery, and stronger
      authored feeling.
- Prefer immersive photography or other concrete visual
      storytelling over empty placeholders, sterile blocks,
       or schematic tidy layouts.
- Do not reward cleanliness, uniformity, or conservative
      structure by themselves if the other side is more
      visually rich, atmospheric, and better aligned to the
       brief.
- Generic landing-page polish, wireframe neatness, or
      conservative regularity should lose to stronger brief
      -specific imagery and concept expression when
      readability remains acceptable.
- You must choose a winner for every item and dimension; do
       not output `tie`.
- If the difference is small, still choose the slightly
      stronger side.
- Base the decision on visible visual quality only.
- If the comparison is close, break the tie in this order:
  1. richer and more convincing imagery
  2. stronger narrative richness and scroll progression
  3. better section-to-section variation and contrast
  4. stronger concept expression and authored atmosphere

Rubric:
{rubric}

Return valid JSON only:
{{
  "overall_winner": "left|right",
  "dimension_winners": {dimension_schema},
  "item_winners": {item_schema},
  "rationale": "<string>"
```

```
}}
"""
```

D.7 Visual-Aesthetic Scoring Anchors

We provide the item-level scoring anchors used by the visual-evaluation prompt. WebJudge scores each visual-aesthetic item on a 7-point ordinal scale: 1 indicates a severe failure of the corresponding visual property, 4 indicates an acceptable but only moderately resolved implementation, and 7 indicates an exceptional, brief-specific, and professionally resolved implementation.

E Additional Experimental Details

This section provides supplementary details for the experimental evaluations. It first describes the human preference evaluation and expert-alignment study used to validate WebJudge and assess perceived webpage quality. It then reports an additional rubric-free generic aesthetic judge study, which tests whether WebWeaver's visual gains extend beyond the VisAWI-inspired WebJudge rubric.

E.1 Human Preference Evaluation and Expert Alignment Study

This subsection provides supplementary details for the human preference evaluation and expert alignment study. The main experiments evaluate whether style-contract construction and propagation improve style-aware prompt-to-webpage generation under the Web4Style/WebJudge setting. The questionnaire interface reported here supports the human preference component of that evaluation.

We recruited 120 Master's and PhD students in computer science and related disciplines from the university community. The participant pool had an approximate male-to-female ratio of 2:1. Participants completed anonymous webpage evaluation tasks involving comparison and ranking of generated webpages under Web4Style briefs. Each session lasted approximately 30 minutes, and participants received compensation of approximately USD 8. No personally identifying information was collected. For the expert alignment study, we recruited 20 participants with relevant professional expertise, including 13 professional web developers and 7 human-computer interaction experts.

We conducted a human preference questionnaire in which each participant was randomly assigned two Web4Style design-brief groups. For each group, participants were shown the original brief and anonymized rendered webpages generated by the compared systems. Participants ranked the webpages under the four VisAWI-Pos dimensions, namely simplicity, diversity, colourfulness, and craftsmanship, as well as overall preference. Both group assignment and within-group webpage order were randomized to reduce ordering bias. Figure 4 shows the questionnaire interface.

The human preference study focuses on visual aesthetics and overall preference and does not directly assess webpage interactivity or usability. Functional validity is evaluated separately by WebJudge through GUI-agent interaction tests, which verify whether the required actions can be successfully executed. Human evaluation of usability, interaction discoverability, and ease of use would provide complementary evidence and is an important direction for future evaluation.

E.2 Rubric-Free Generic Aesthetic Judge

To assess whether WebWeaver merely aligns with the VisAWI-inspired WebJudge rubric, we conduct an additional rubric-free aesthetic evaluation using generic visual judges. Unlike WebJudge, these evaluators are not given any VisAWI dimensions, scoring rubrics, or aesthetic criteria. Instead, they receive only the original design brief and two anonymized rendered webpage screenshots generated for the same brief.

The evaluators are asked the following generic question: Which webpage is better overall in terms of professional visual quality, visual coherence, and suitability for the design brief? The full evaluation prompt is shown below.

```
You are evaluating two rendered webpages generated from the
      same design brief.

Design brief: \{brief\}

Webpage A: \{image A\}

Webpage B: \{image B\}

Which webpage is better overall in terms of professional
     visual quality, visual coherence, and suitability for
      the design brief?

Choose exactly one: A or B. Briefly explain your choice in
     one sentence.
```

We use both `Gemini-2.5-Flash` and `gpt-4o-mini` as independent generic aesthetic judges. Screenshots are anonymized before evaluation, and webpage ordering is randomized

Table 10: WebJudge visual-aesthetics scoring anchors for Simplicity.

Item	Score	Anchor
Clean layout	1	Visually cluttered; elements overlap or compete heavily; spacing and grouping are poor.
	2	Cluttered and noisy; some grouping exists, but layout tension and spacing issues are obvious.
	3	Below clean; several crowded areas, uneven spacing, or distracting visual leftovers remain.
	4	Mostly clean but with ordinary spacing discipline; some clutter or unresolved density remains.
	5	Clearly clean and readable, but still somewhat conventional, schematic, or limited in richness.
	6	Very clean and polished with deliberate spacing and clear grouping, while still supporting a rich authored presentation.
	7	Exceptionally clean and precisely resolved without becoming sterile; whitespace, grouping, and pacing feel unusually intentional for this brief.
Easy to grasp	1	Hard to tell what the page is for or where the eye should go first.
	2	Purpose is weakly signaled; understanding requires effort and scanning.
	3	Basic purpose is inferable, but hierarchy is muddled or delayed.
	4	Purpose becomes understandable after a moment; hierarchy is only moderately clear.
	5	Generally easy to understand; primary message and flow are clear with small friction points.
	6	Very easy to grasp; hierarchy is strong and user attention is guided well.
	7	Immediately understandable; hierarchy feels effortless, intentional, and highly polished.
Well structured	1	Sections feel random; alignment or grid logic is inconsistent; structure is confusing.
	2	Some major structure exists, but section sequencing or grid logic is weak.
	3	Recognizable structure exists, but important layout relationships feel loose or inconsistent.
	4	Basic structure exists, but layout logic is not fully consistent or refined.
	5	Well structured overall, with only minor issues in sequencing, grouping, or alignment logic.
	6	Strong structure with deliberate sectioning and reliable layout logic throughout most of the page.
	7	Information architecture is exceptionally clear; section flow and grid feel refined throughout.
Uniform	1	Component styles vary unpredictably; buttons, cards, spacing, or typography feel mismatched.
	2	Some shared styling exists, but inconsistencies are frequent and distracting.
	3	Moderate consistency, but several components break the visual system.
	4	Some consistency, but a few components noticeably diverge from the system.
	5	Mostly uniform; the system is clear, though some component decisions still feel generic, repetitive, or only partly resolved.
	6	Highly consistent across components and spacing, with only very minor deviations and without collapsing into template sameness.
	7	Exceptionally uniform and expressive; the design language stays coherent, rich, and deliberate across the full page.
Coherent	1	Themes, typography, colors, or imagery feel mismatched and conceptually disconnected.
	2	A partial theme exists, but multiple elements still feel out of place.
	3	Some coherence is present, though several design decisions do not fully fit together.
	4	Generally cohesive, but some elements still feel stylistically out of place.
	5	Coherent overall, though concept expression or mood fit still feels somewhat safe or only partly developed.
	6	Very cohesive; typography, color, imagery, and layout strongly reinforce the requested concept and mood.
	7	Exceptionally unified and brief-specific; the full page expresses one clear concept with no visible thematic drift.

to reduce positional bias. We report win rate, 95% bootstrap confidence intervals, and a two-sided binomial test against the null hypothesis that WebWeaver is preferred with probability 0.5.

The rubric-free generic aesthetic evaluation shows that WebWeaver remains consistently preferred even when the evaluators are not exposed to VisAWI-inspired dimensions or the WebJudge rubric. As shown in Table 14, across both `gpt-4o-mini` and `Gemini-2.5-Flash`, WebWeaver achieves strong preference margins over OpenHands, Bolt.diy, and WebGen-Agent under a generic perceptual evaluation setting. Although the gains over the vanilla direct-generation baseline are smaller and less stable, the overall results suggest that WebWeaver improves broader perceptual webpage quality beyond rubric-specific optimization.

Figure 5 shows a representative preference difference case between WebJudge and the rubric-free generic aesthetic evaluation.

E.3 Reproducibility and Execution Settings

Each system was run once per Web4Style task under its intended workflow. WebWeaver used its validation-and-repair loop and required, on average, 22k tokens and 412.3 s per webpage. The direct LLM baseline generated each webpage once without tools or repair, averaging 1.1k tokens and 7.5 s. OpenHands, Bolt.diy, and WebGen-Agent used their native workflows and supported repair, averaging 77k tokens/236 s, 13k tokens/78 s, and 21k tokens/156 s per webpage, respectively. Unless otherwise specified, generation used provider-default settings. We report model/API versions, access dates, and any overridden decoding parameters in the released configuration. WebJudge used fixed prompts with a maximum output length of 1,000 tokens. Browser-based evaluation used Selenium and Google Chrome, with server readiness checked every 2 s for up to 30 s.

Table 11: WebJudge visual-aesthetics scoring anchors for Diversity.

Item	Score	Anchor
Interesting	1	Plain or boring; looks like an unstyled template or default scaffold.
	2	Slightly more than bare-bones, but still largely uninteresting and generic.
	3	A few engaging touches exist, but the page remains mostly standard or forgettable.
	4	Some engaging elements are present, but the overall impression is still fairly standard.
	5	Clearly interesting in places, though parts of the page still feel familiar, repetitive, or like a polished template.
	6	Strongly engaging and visually compelling, with sustained interest across sections and meaningful imagery or treatment contrast.
	7	Memorable, compelling, and rare; the page stands out through authored imagery, pacing, and brief-specific atmosphere.
Inventive	1	No creative layout or presentation choices; entirely predictable.
	2	Token attempts at creativity, but mostly conventional and unconvincing.
	3	Some creative ideas are visible, though they feel limited or only partly realized.
	4	Minor creative touches, but the page remains mostly conventional.
	5	Shows clear creative thinking in layout, imagery orchestration, section emphasis, or concept expression, though some parts still rely on familiar solutions or repeated treatments.
	6	Strongly inventive and well executed, with distinctive visual decisions, authored pacing, or richer concept-led presentation across the page, even if the core structure remains readable and not radically unconventional.
	7	Distinctive and creative in a convincing, fully resolved way; the inventiveness feels rare, visual, and essential to the brief through authored presentation, storytelling, and image-aware composition.
Inspired	1	Feels unconsidered; the aesthetics seem accidental rather than designed.
	2	Some taste is visible, but the page still feels underdirected or weakly authored.
	3	Design intent exists, but polish and originality are limited.
	4	Some design intent is visible, but refinement or originality remains moderate.
	5	Clearly designed with taste, though the art direction or authorship still feels limited, safe, or only partly immersive.
	6	Strong sense of design intent and refinement, with clear authorship, mood, and visual storytelling across sections.
	7	Strong design intent and taste are unmistakable; the page feels genuinely art-directed, immersive, and rare in its level of refinement.
Dynamic	1	Static and flat; poor use of scale, contrast, or directional flow.
	2	Slight variation exists, but the page still feels visually inert.
	3	Some motion or rhythm is implied, but the composition still lacks energy.
	4	Some variation in scale or sections, but the overall energy is only moderate.
	5	Good visual momentum, with a readable sense of progression or pacing, though some stretches still feel familiar, repeated, or less energized.
	6	Strong rhythm and flow create clear visual movement, scroll progression, or narrative energy through pacing, imagery, and emphasis changes, without requiring extreme layout shifts.
	7	Highly dynamic and expertly controlled; rhythm, imagery, scale, and narrative progression sustain authored energy across the full page.
Pleasantly varied	1	Monotonous repetition or chaotic variation makes the page unpleasantly uneven.
	2	Variation is weak or awkward; sections feel too samey or inconsistently different.
	3	Some variation exists, but repetition or inconsistency remains noticeable.
	4	Moderate variation, though the page is a bit repetitive or slightly inconsistent.
	5	Pleasant variety across sections, though some treatments still feel repeated, safe, or not fully differentiated.
	6	Rich and controlled variation; sections feel distinct, image-aware, paced, and cohesive throughout the page.
	7	Exceptionally varied in a purposeful, elegant way; each section adds fresh visual value without feeling random.

F Additional Evaluation on WebGen-Bench

To examine whether WebWeaver's gains transfer beyond Web4Style and the WebJudge evaluation protocol, we conduct an additional evaluation on WebGen-Bench (Lu et al., 2025b). We sample 20 tasks from WebGen-Bench and evaluate all methods under the benchmark's original evaluation protocol. For a controlled comparison, we use DeepSeek-V3 as the common backbone. We compare WebWeaver against the vanilla LLM baseline, OpenHands, Bolt.diy, and WebGen-Agent.

Table 15 shows that WebWeaver achieves the best performance on both metrics, with 61.8% Accuracy and a 4.3/5 Appearance Score. Compared with the vanilla LLM baseline, WebWeaver improves Accuracy by 35.3 percentage points and Appearance Score by 3.1 points, suggesting that explicit style-contract propagation benefits both task completion and visual quality. WebWeaver also outperforms WebGen-Agent, the strongest baseline, by 9.2 percentage points in Accuracy and 0.5 points in Appearance Score. These results suggest that the gains are not limited to Web4Style or the WebJudge protocol.

G Additional Case Studies

To complement the main text, we provide qualitative case studies across different brief types, style regimes, and interaction patterns. Each example follows the style-contract lifecycle used throughout the paper: the original Web4Style brief is first grounded into a design context; WebWeaver then constructs a style contract; the contract is propagated into page planning, visual assets, and front-end implementation; and the final rendered webpage is inspected for coherence with the brief and contract. The resulting case studies are shown in Figures 6, 7, 8, and 9.

Table 12: WebJudge visual-aesthetics scoring anchors for Colourfulness.

Item	Score	Anchor
Attractive composition	1	Palette is unpleasant, discordant, muddy, or visibly harmful to quality.
	2	Some useful color exists, but composition still feels awkward or weakly balanced.
	3	Serviceable but clearly flawed; contrast or palette relationships have noticeable weaknesses.
	4	Acceptable palette, though not especially attractive; minor contrast or balance issues may exist.
	5	Attractive overall with solid palette relationships; only limited refinement is missing.
	6	Strong and polished color composition with confident contrast hierarchy and mood control.
	7	Highly refined palette and contrast hierarchy; color composition feels expertly judged.
Perfect color choice	1	Colors feel wrong for the context and distract from readability or brand fit.
	2	Some colors fit, but key choices still feel off for the brief or mood.
	3	Reasonable intent is visible, but several colors are weakly tuned to the context.
	4	Colors are reasonable, though not especially well tuned to the brand or purpose.
	5	Color choices fit the brief well overall, with only minor mismatches or missed nuance.
	6	Color choices are strongly aligned to the brief, mood, and brand, with small remaining limitations.
	7	Color choices are exceptionally well matched to the brief, mood, and brand context.
Colors match	1	Colors clash noticeably; accents, backgrounds, and typography do not work together.
	2	Some color relationships work, but multiple mismatches remain visible.
	3	Mostly workable, though a few pairings feel off or awkward.
	4	Mostly matching, but a few mismatches still stand out.
	5	Colors match well overall, though the palette still relies on safe or predictable pairings.
	6	Colors work together very smoothly; pairings feel deliberate, nuanced, polished, and supportive of the intended mood.
	7	Colors work together seamlessly and characterfully; every pairing feels precise, expressive, and unmistakably tied to the brief.
Appealing colors	1	Colors are harsh, dull, or washed out enough to reduce perceived quality.
	2	Slightly appealing in places, but overall still unrefined or weak.
	3	Some appealing tones are present, though the palette remains ordinary or inconsistent.
	4	Colors are acceptable but not especially memorable or rich.
	5	Appealing overall; the palette adds quality, though depth or distinctiveness is limited.
	6	Clearly appealing and polished with strong tonal control and perceived quality.
	7	Notably appealing and polished; the colors materially elevate the page.

For each case, the left column gives the Web4Style brief, while the right column explains how WebWeaver grounds the brief, constructs a style contract, and propagates the contract into the final webpage. These examples are qualitative illustrations of the mechanism studied quantitatively in Section 7; they are not additional independent evaluations.

H Failure Cases

We further inspect representative failure cases from WebWeaver-generated webpages. Figures 10 and 11 show two examples covering style realization and layout fidelity.

Case 000046 shows that an explicit style contract does not guarantee full realization of high-level visual mechanisms. WebWeaver preserves the intended cinematic production identity and produces a coherent visual composition, but it does not implement the requested reel-driven hero behavior. This suggests that dynamic or media-specific style requirements require stronger enforcement during implementation and validation.

Case 000036 illustrates a layout-level fidelity error. The generated page follows the general editorial portfolio direction, but misses the requested full-bleed, vertically stacked project rhythm. Instead, it uses a more standard grid structure, indicating that some spatial constraints may be softened when translated from style contract to page layout.

Table 13: WebJudge visual-aesthetics scoring anchors for Craftsmanship.

Item	Score	Anchor
Professionally designed	1	Looks amateur; alignment, typography, and spacing feel obviously unpolished.
	2	Some competent parts exist, but overall presentation still feels amateurish.
	3	Moderately competent, but rough edges are noticeable and reduce professional credibility.
	4	Serviceable but with visible rough edges; professional quality is only partial.
	5	Clearly professional overall, though some resolutions still feel generic, familiar, schematic, or less than premium.
	6	Strongly professional and polished, with clear authorship, richer execution, and only minor issues preventing premium quality.
	7	Convincingly professional at a premium, rare standard; the page feels distinctly authored, immersive, and far beyond mere competence.
Up to date	1	Feels outdated; relies on default styles or old-fashioned UI conventions.
	2	A few modern touches exist, but the page still feels noticeably dated.
	3	Some current patterns are used, but the experience remains only partly modern.
	4	Some modern patterns are present, but not consistently applied.
	5	Feels current overall, though still somewhat ordinary, overfamiliar, or not especially sophisticated in execution.
	6	Clearly modern and polished with only small traces of dated or overfamiliar execution, and without leaning on empty template polish.
	7	Sophisticated, current, and unusually well judged; it feels contemporary through rich execution, not just default modern styling.
Designed with care	1	Many small issues accumulate; details feel careless or unresolved.
	2	Some care is visible, but too many alignment, spacing, or typography issues remain.
	3	Moderate care is visible, though several inconsistencies still weaken the page.
	4	Some details are handled, but multiple inconsistencies remain.
	5	Designed with care overall, though several details still feel formulaic, repeated, or only partly resolved.
	6	Careful and polished in detail; most micro-decisions feel deliberate, brief-aware, and integrated into a richer page concept.
	7	Fine details are consistently polished and purposeful; craftsmanship is obvious even on close inspection and reinforces the authored concept.
Clear concept	1	No clear theme or concept; visuals do not support an intended identity.
	2	A weak concept is implied, but it is vague, inconsistent, or under-expressed.
	3	A concept exists, but execution is incomplete or uneven enough to weaken clarity.
	4	A concept is present but not strongly expressed or consistently executed.
	5	Clear concept overall; it is readable and coherent, though still somewhat safe, generic, or only moderately distinctive.
	6	Strong concept expression with consistent execution, richer storytelling, and only minor limitations in distinctiveness or depth.
	7	Distinctive, consistent, and powerfully expressed across the full page; the concept feels unmistakably tied to the brief and visually immersive.

Table 14: Rubric-free generic aesthetic judge results using two independent VLM judges. The evaluators are not given VisAWI-style dimensions or the WebJudge rubric.

Judge	Comparison	Win Rate (95% CI)	p
GPT-4o-mini	WebWeaver vs Vanilla LLM	61.36% [47.73%, 72.73%]	0.1797
	WebWeaver vs OpenHands	90.91% [81.82%, 97.73%]	< 0.001
	WebWeaver vs Bolt.diy	75.00% [61.36%, 88.64%]	0.0074
	WebWeaver vs WebGen-Agent	84.09% [75.00%, 93.18%]	< 0.001
Gemini-2.5-Flash	WebWeaver vs Vanilla LLM	54.55% [38.64%, 68.24%]	0.7744
	WebWeaver vs OpenHands	95.45% [88.64%, 100.00%]	< 0.001
	WebWeaver vs Bolt.diy	68.18% [54.55%, 81.82%]	0.0386
	WebWeaver vs WebGen-Agent	81.82% [72.73%, 90.91%]	< 0.001

Table 15: Additional evaluation on 20 WebGen-Bench samples using DeepSeek-V3 as the common backbone.

Method	Accuracy (%) ↑	Appearance Score ↑
Vanilla LLM	26.5	1.2
OpenHands	9.0	1.5
Bolt.diy	20.8	2.0
WebGen-Agent	52.6	3.8
WebWeaver	**61.8**	**4.3**

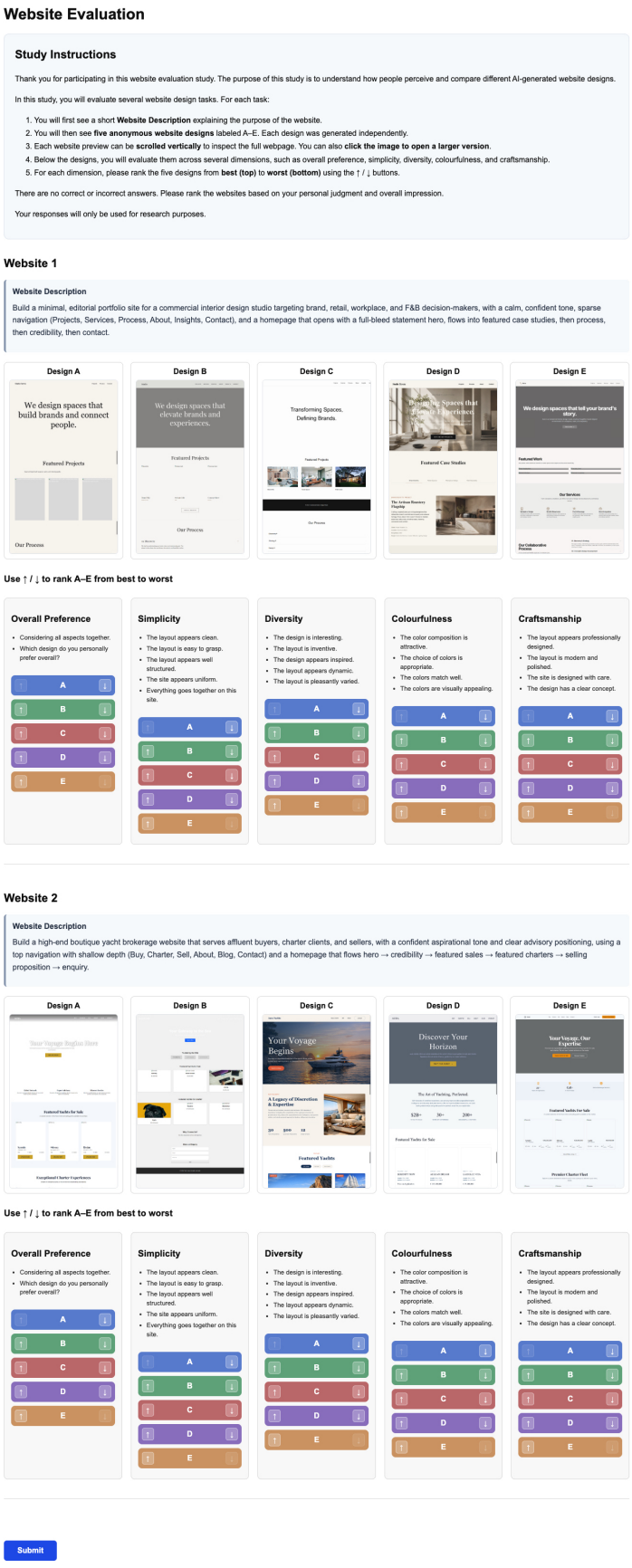


Figure 4: Human preference questionnaire interface. Participants rank anonymized rendered webpages generated for the same Web4Style brief under the four VisAWI-Pos dimensions and overall preference.

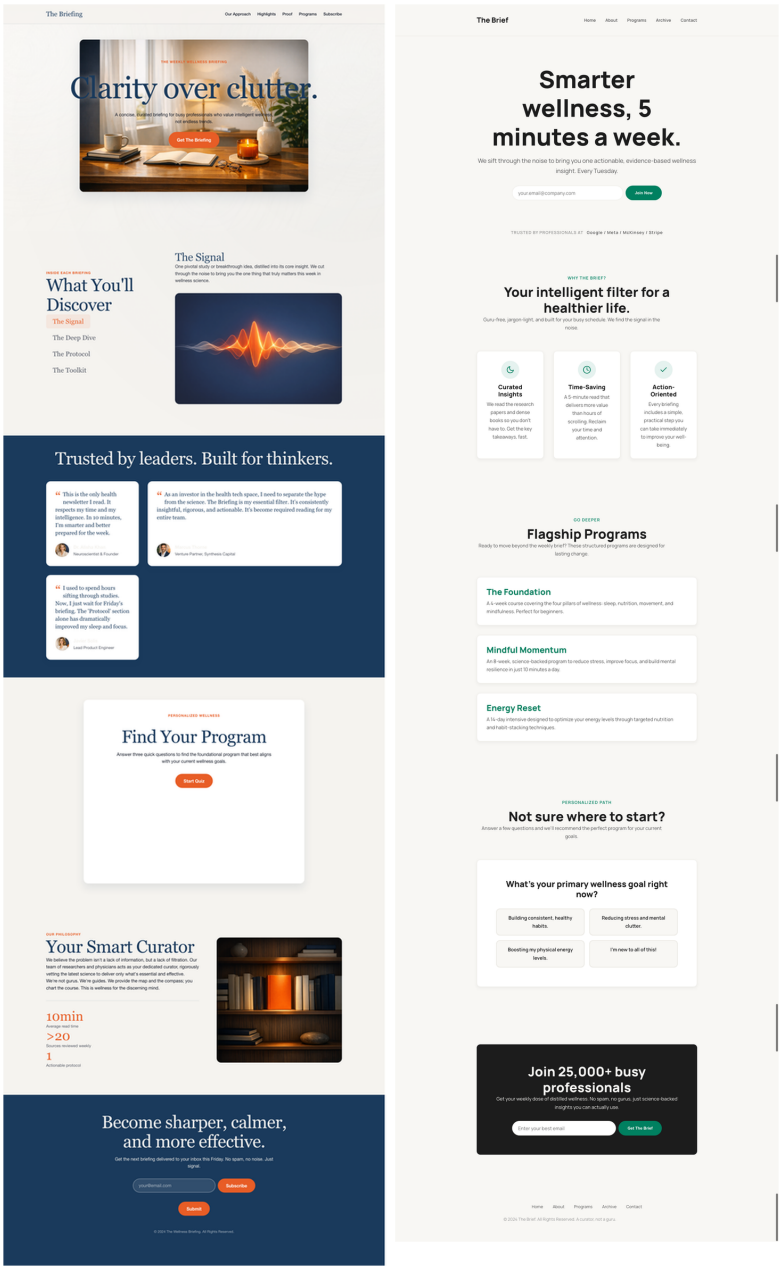


Figure 5: Case study from the rubric-free generic aesthetic evaluation (Case 000076). The left webpage is generated by WebWeaver, while the right webpage is generated by the vanilla LLM baseline. In this example, WebJudge prefers the WebWeaver result, while the generic judge prefers the vanilla baseline. The two evaluators emphasize different aspects of webpage quality, including visual storytelling and concise presentation style.

Sample 000001

Brief	Case interpretation
Build a minimal, editorial portfolio site for a commercial interior design studio targeting brand, retail, workplace, and F&B decision-makers, with a calm, confident tone, sparse navigation (Projects, Services, Process, About, Insights, Contact), and a homepage that opens with a full-bleed statement hero, flows into featured case studies, then process, then credibility, then contact. Use large typographic hierarchy with generous white space, restrained color (off-white, deep charcoal, one accent), subtle material cues via photography only, and quiet micro-interactions with smooth, low-friction scrolling reveals. Prioritize project grids, process accordion, and inquiry form, exclude blogs, logins, and e-commerce, and make every interaction drive viewing projects, understanding the process, and initiating contact.	This example shows how WebWeaver handles a minimal editorial portfolio brief for a commercial interior design studio targeting brand, retail, workplace, and F&B decision-makers. The case requires the system to preserve a calm architectural identity while still supporting project browsing, process explanation, and inquiry conversion. WebWeaver grounds the brief into a studio-positioning context and constructs an *Editorial Architectural* style contract: warm off-white and charcoal tones, restrained accent use, serif-led hierarchy, generous whitespace, quiet motion, and disciplined project-grid behavior. The PLAN AGENT then propagates this contract into a page sequence covering studio introduction, featured case studies, process, credibility, and contact. This sample is useful for examining whether a sparse editorial brief can remain coherent as the style contract is carried through planning, asset generation, and front-end realization. See Figure 6 for more details.

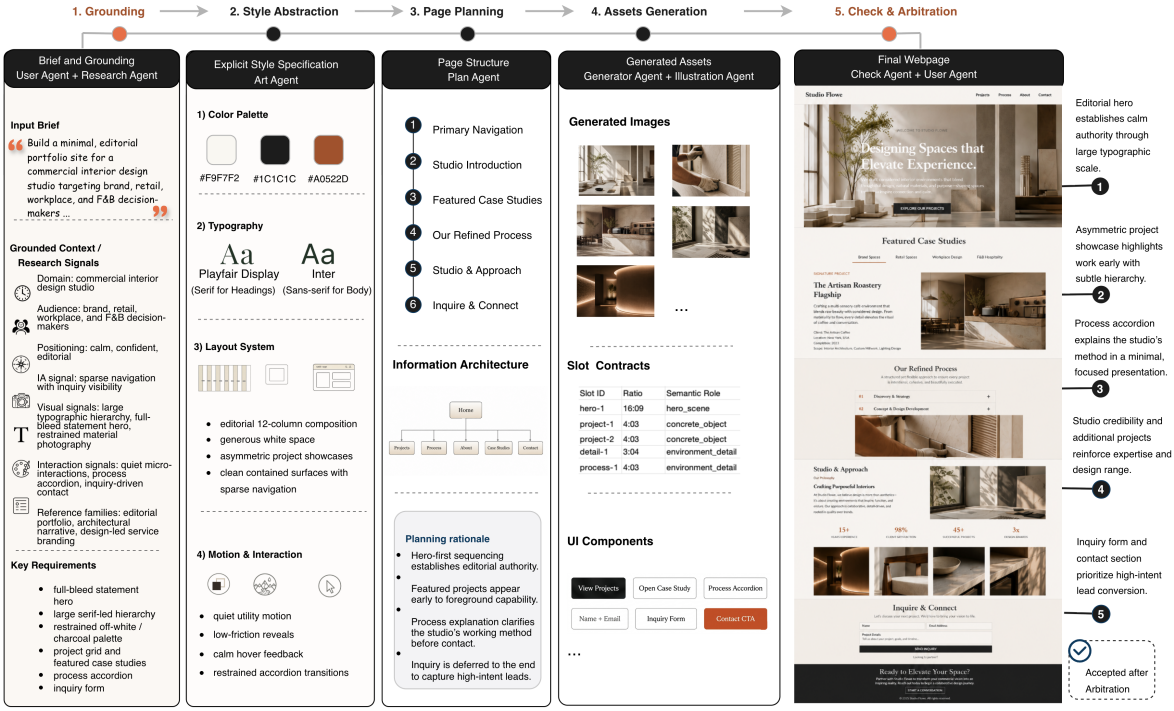


Figure 6: Appendix case study from Sample 000001. WebWeaver constructs an editorial architectural style contract and propagates it into a sparse portfolio webpage with controlled hierarchy, project browsing, and inquiry-oriented interaction.

Sample 000011

Brief	Case interpretation
Build a high-end boutique yacht brokerage website that serves affluent buyers, charter clients, and sellers, with a confident aspirational tone and clear advisory positioning, using a top navigation with shallow depth (Buy, Charter, Sell, About, Blog, Contact) and a homepage that flows hero → credibility → featured sales → featured charters → selling proposition → enquiry. Use full-bleed photography with tight focal points, strong typographic hierarchy (XL hero, clear card titles, restrained body), generous white space, and a cool, nautical color palette with high-contrast CTAs. Implement card-based listings, multi-step enquiry forms, subtle hover and fade transitions, and micro-interactions on filters, while excluding user logins, online payments, and self-serve dashboards; prioritize form submissions, phone/WhatsApp taps, and "custom request" actions as primary conversions.	This example shows how WebWeaver handles a high-end yacht-brokerage brief that combines aspirational visual identity with conversion-oriented functionality. The case requires the system to coordinate editorial credibility, sale and charter listings, and broker-led inquiry without losing a luxury advisory tone. WebWeaver grounds the brief into a brokerage context and constructs a *Yachting Elegance* style contract: deep nautical contrast, cinematic full-bleed imagery, expressive serif headlines, polished listing cards, and restrained transition behavior. The contract is propagated into a page plan that moves from hero and credibility to featured yachts, seller proposition, and contact. This sample is useful for examining whether style-contract propagation can keep a luxury conversion flow visually coherent across listing modules, inquiry components, and final implementation. See Figure 7 for more details.

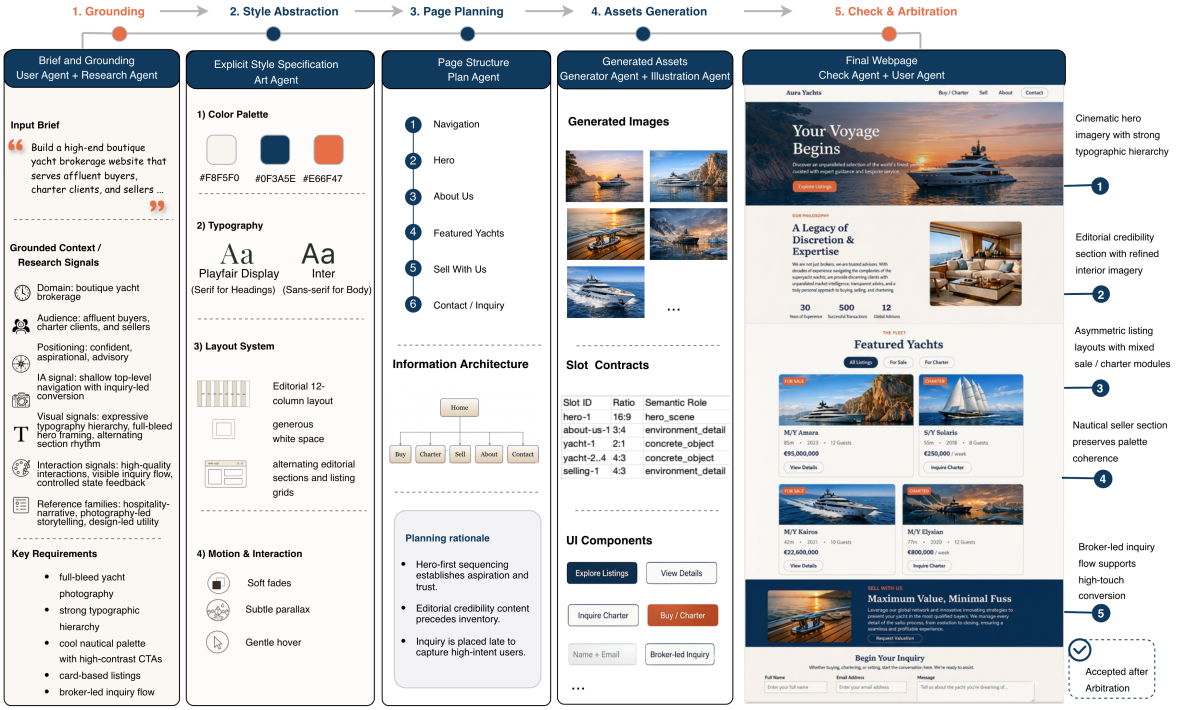


Figure 7: Appendix case study from Sample 000011. WebWeaver constructs a luxury yacht-brokerage style contract and propagates it into coordinated imagery, listing modules, and inquiry-oriented interaction.

Sample 000051

Brief	Case interpretation
Build a premium ecommerce and brand-story site for a heritage beverage roaster, aimed at consumers, gift buyers, and hospitality professionals, with a calm, confident tone and craftsmanship positioning, using a persistent mega-nav that separates Shop, Brand World, Experiences, Professionals, Magazine, and Contact into shallow, scannable levels. The homepage should open with a full-bleed hero and primary CTA, flow into carousels of best sellers and special editions, then a guided "taste profiler," category gateways, lifestyle accessories and gifts, B2B solutions, training/experience highlights, latest articles, and a newsletter block, all supported by rich photography, restrained iconography, and a clear card hierarchy with generous whitespace, large serif/sans pairing, a tight type scale, warm neutral palette with a single accent, soft shadows, subtle parallax, low-friction microinteractions, simple cart/account/search, and no community forums or user-generated galleries.	This example shows how WebWeaver handles a premium ecommerce brief that must balance product merchandising, brand storytelling, and professional-service pathways within one coherent surface. The case requires the system to coordinate best-seller carousels, a guided taste-profiler interaction, category gateways, B2B entry points, editorial content, and newsletter conversion without collapsing into a generic storefront. WebWeaver grounds the brief into a heritage beverage context and constructs an *Editorial Heritage* style contract: warm tactile neutrals, a single energetic accent, serif-led editorial hierarchy, asymmetric grids, rich product imagery, and low-friction commerce interactions. The resulting page plan moves from navigation and hero through curated favourites, guided profiling, category exploration, partner-facing content, journal-style storytelling, and contact. This sample shows how style-contract propagation allows ecommerce structure to remain aligned with a calm heritage brand language. See Figure 8 for more details.

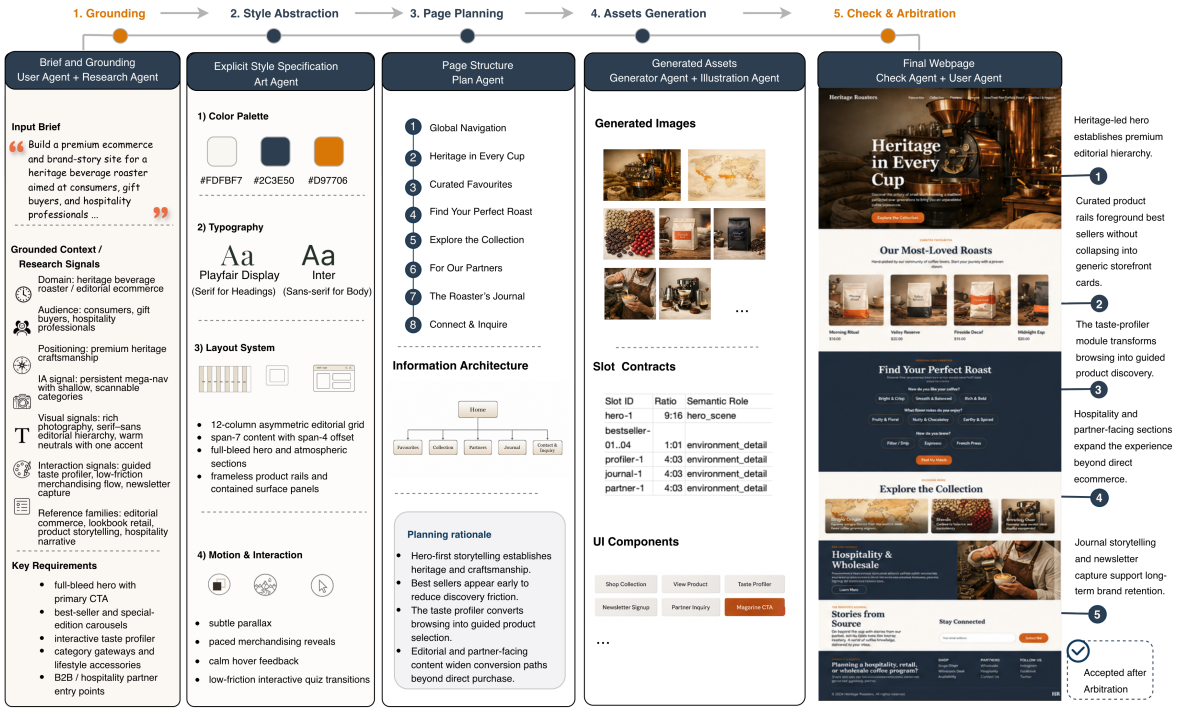


Figure 8: Appendix case study from Sample 000051. WebWeaver constructs an editorial heritage style contract and propagates it into coordinated merchandising, guided product discovery, brand storytelling, and front-end implementation.

Sample 000081

Brief	Case interpretation
Build a high-end digital product studio marketing site for design- and engineering-led teams, with a single, deep-scrolling marketing homepage plus focused Work, Services, About, and Contact sections, using a confident, minimal, premium tone positioned as senior, strategic partners. Use a split light/dark aesthetic with strong contrast, large typographic hero, generous white space, and clear visual hierarchy, layering case-study visuals, product shots, and subtle 3D or motion accents revealed via smooth, physics-based scroll transitions, pinned segments, and micro-interactions. Prioritize portfolio browsing, service understanding, and contact initiation, exclude blogs and logins, and implement responsive fluid typography, modular grids, soft shadows, and restrained animations that keep performance high while guiding users to "View work," "Explore services," and "Start a project" CTAs.	This example shows how WebWeaver handles a premium digital-product-studio brief with strong motion and narrative-scroll requirements. The case requires the system to maintain a sparse, high-contrast strategic identity while staging case-study visuals, service explanations, and project-initiation paths through a deep-scrolling page. WebWeaver grounds the brief into a senior product-studio context and constructs a *Strategic Contrast* style contract: alternating deep-ink and paper-white sections, oversized display typography, asymmetric modular compositions, crisp component behavior, and restrained motion accents. The PLAN AGENT propagates the contract into a page flow from hero through selected work, services, process, studio positioning, and contact. This sample is useful for examining how motion-aware design intent is carried through style-contract construction, structural planning, and front-end realization. See Figure 9 for more details.

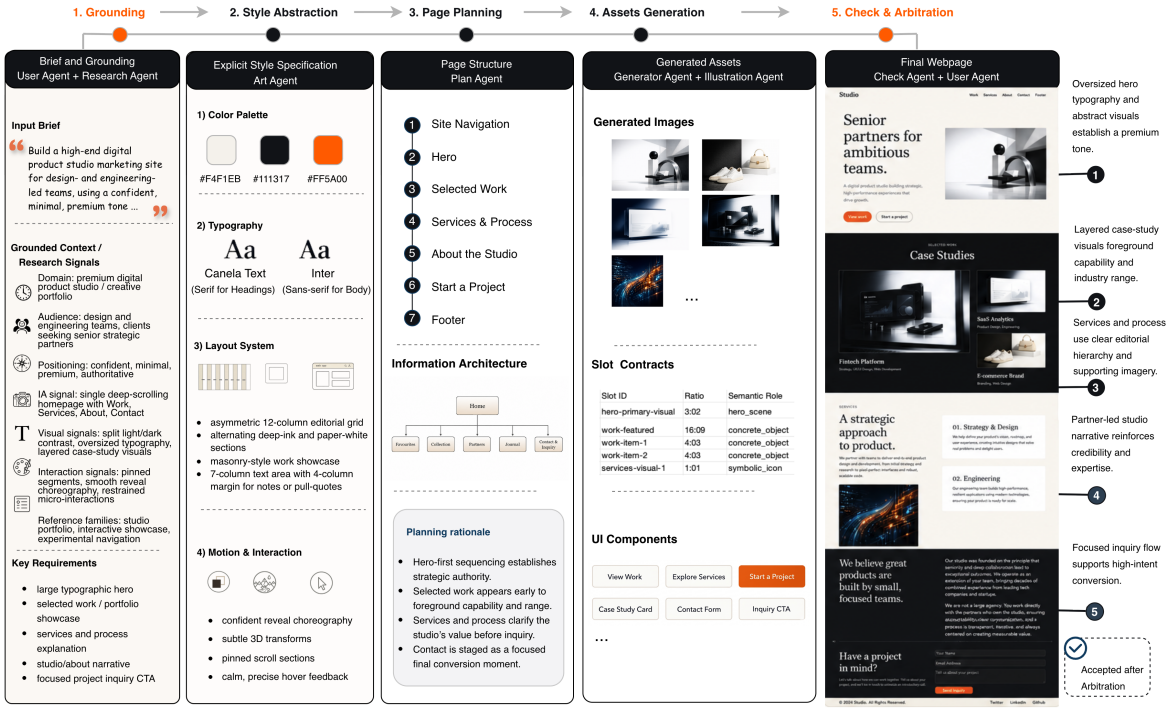


Figure 9: Appendix case study from Sample 000081. WebWeaver constructs a strategic-contrast style contract and propagates it into motion-aware pacing, portfolio storytelling, service explanation, and contact conversion.

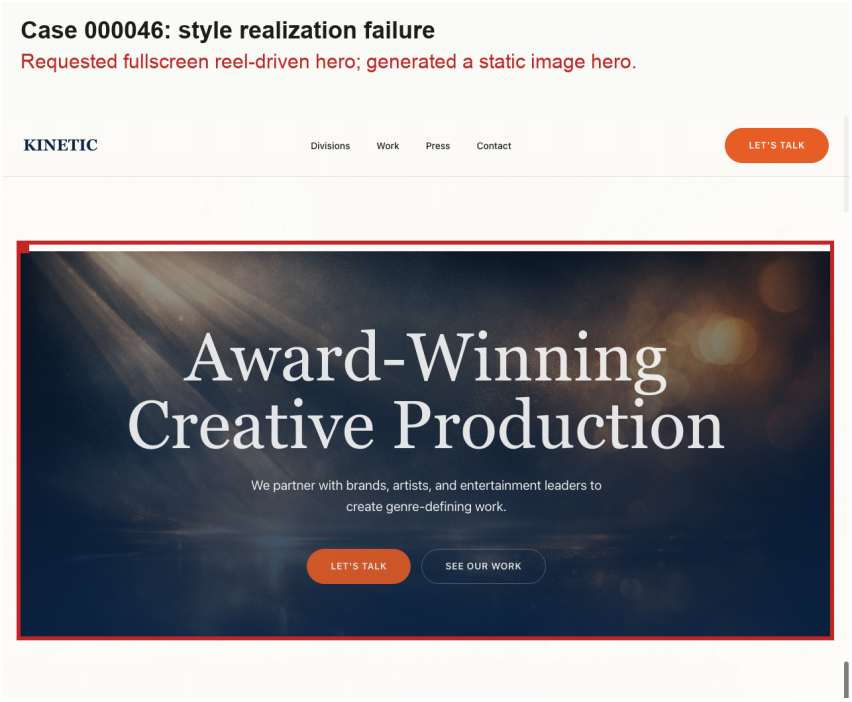


Figure 10: Style realization failure in Case 000046, generated by WebWeaver. The brief requests a fullscreen reel-driven hero, but WebWeaver realizes this requirement as a static image hero. Although the page remains visually polished, the generated result misses the requested dynamic media treatment.

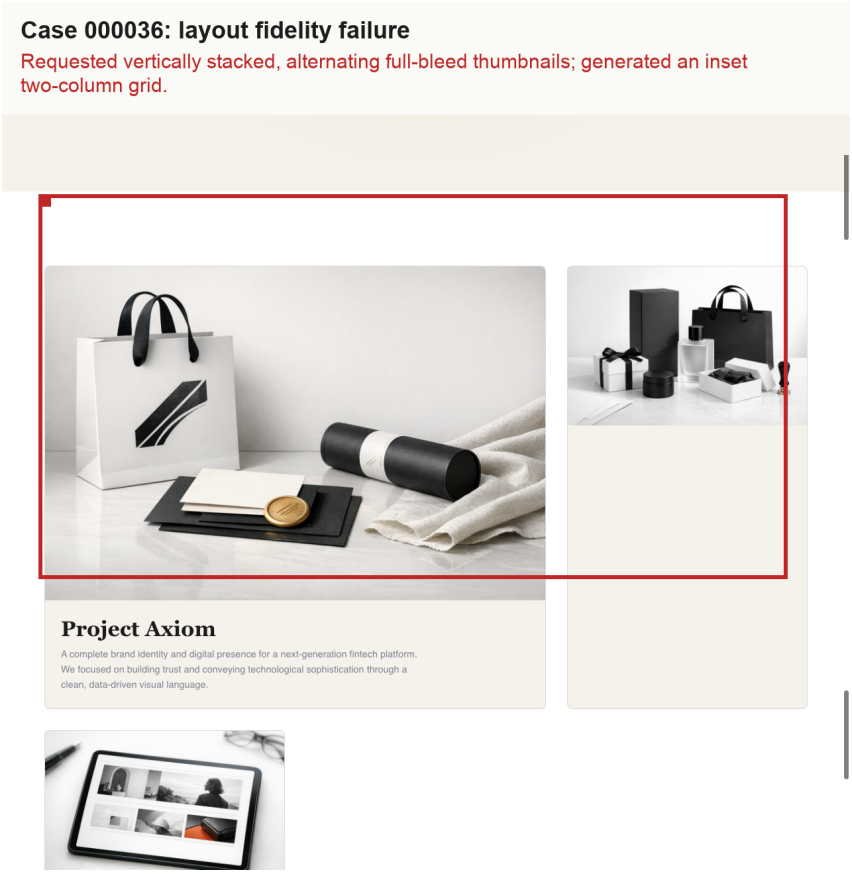


Figure 11: Layout fidelity failure in Case 000036, generated by WebWeaver. The brief requests vertically stacked, alternating full-bleed case thumbnails, but the generated page falls back to a conventional inset two-column grid. This shows that layout-specific style constraints can be weakened during propagation from the style contract to frontend realization.