WebMMU: A Benchmark for Multimodal Multilingual Website Understanding and Code Generation

Rabiul Awal[1,2,3*] **Mahsa Massoud**[1,2,4*] **Aarash Feizi**[1,2,4†] **Zichao Li**[1,2,4†]
Suyuchen Wang[1,2,3] **Christopher Pal**[1,2,6] **Aishwarya Agrawal**[2,3] **David Vazquez**[1]
Siva Reddy[1,2,4] **Juan A. Rodriguez**[1,2,5] **Perouz Taslakian**[1]
Spandana Gella[1] **Sai Rajeswar**[1,2]
[1]ServiceNow, [2]Mila, [3]Université de Montréal, [4]McGill University,
[5]École de Technologie Supérieure (ETS), [6] Polytechnique Montréal

Abstract

We present WebMMU, a multilingual benchmark that evaluates three core web tasks: (1) website visual question answering, (2) code editing involving HTML/CSS/JavaScript, and (3) mockup-to-code generation. Unlike prior benchmarks that treat these tasks separately, WebMMU unifies them using expert-annotated, real-world web data to assess models' abilities in complex multi-step reasoning, precise element grounding, and functional UI comprehension and coding. Our evaluation shows that while multimodal large language models (MLLMs) perform well on basic information extraction, they struggle with reasoning and grounding, editing code to preserve functionality, and generating design-to-code that maintains hierarchy and supports multilingual content. These findings reveal key limitations in current MLLMs and underscore the need for improved multimodal and cross-lingual reasoning to build future web agents capable of automating diverse web development tasks. The dataset is publicly available at webmmu-paper.github.io.

1 Introduction

The web is vital to daily life, enabling information access, shopping, and communication. Multimodal large language models (MLLMs) (Wang et al., 2024b; Hurst et al., 2024) that understand the *Visual Web* can help users extract information, support tasks like budget-conscious shopping, and handle multiple languages (Deng et al., 2024). They also show promise in automating web design and development, including front-end layout creation, user interface (UI) editing, and code generation (Anthropic, 2024). Unlike tasks focused only on text or images (Wang et al., 2024c; Yue et al., 2024), visual web understanding requires combining UI structure, layouts, text, interactivity, and visuals.

Existing benchmarks target specific aspects of web tasks but remain fragmented and lack comprehensive coverage. Website VQA datasets like WebQA (Chang et al., 2022) and WebSRC (Chen et al., 2021b) mainly focus on text retrieval, overlooking reasoning over UI structure, interactivity, and multilingual content. Recent web agent benchmarks evaluate online task completion (Koh et al., 2024; Deng et al., 2024; He et al., 2024), showing promise for agentic AI but are limited to artificial websites or lack fine-grained categorization (e.g. grounding, understanding, multi-step reasoning). In web development, design-to-code datasets such as Pix2Code (Beltramelli, 2018) and Web2Code benchmarks (Yun et al., 2024), as well as sketch-based datasets like Sketch2Code (Li et al., 2024b), cover a limited variety of UIs and often fail to capture real web variability due to automated creation. Furthermore, current benchmarks lack multilingual and cross-domain generalization, limiting applicability beyond English and specific domains. These gaps motivate a unified benchmark integrating multiple web tasks with multimodal, reasoning, and cross-lingual capabilities for effective evaluation of AI in web development and advanced web understanding.

To address these challenges, we introduce **WebMMU** (Figure 1), a multimodal, **M**ultilingual, and **MU**lti-task benchmark for evaluating MLLMs on the Visual **Web** in four languages: English, Spanish, German, and French. It spans 20 real-world website domains such as *shopping*, *booking*, *sports*, and *technology*. WebMMU covers three core tasks: **Website VQA (WebQA)**, which tests functional understanding, visual comprehension, and multi-step reasoning via visual question-answering; **Mockup2Code Generation**, assessing design-to-code alignment for UI mockups and sketches, including both simple and complex nested lay-

*Co-first authors † Co-second authors

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 25130–25157
November 4-9, 2025

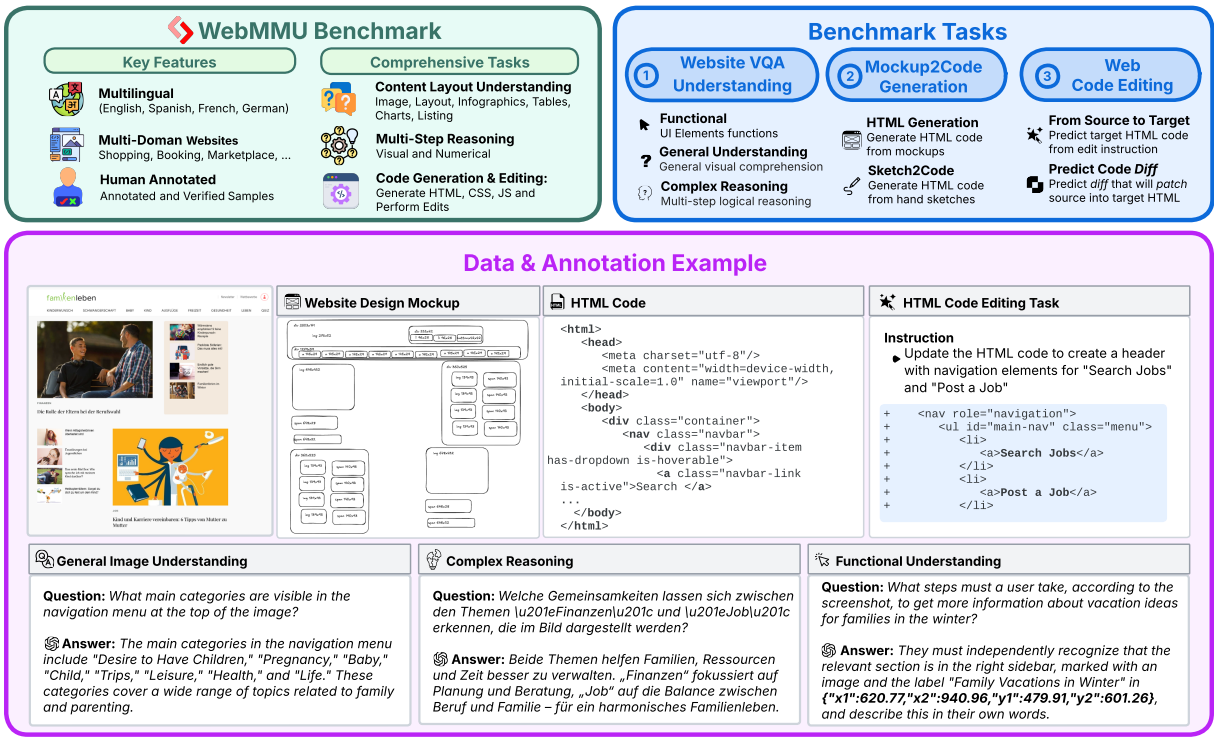


Figure 1: **WebMMU Benchmark Overview.** WebMMU evaluates models on diverse web-based tasks: WebQA, Mockup2Code, and Web Code Editing. Covering 20 domains and four languages, it challenges models to answer visual questions requiring multi-step reasoning and action grounding. It also assesses design-to-code generation from website layouts of varying complexity and evaluates code editing for automated web development.

outs; and **Web Code Editing**, evaluating precise, context-aware HTML/CSS/JavaScript code editing for feature additions, UI tweaks, and bug fixes.

We benchmark state-of-the-art MLLMs across three core tasks, evaluating both open-source and closed-source models. Our results reveal significant challenges in action grounding and complex reasoning in the WebQA task, along with difficulties in structured layout understanding and accurate code generation for web development. While models (in particular, closed-source ones) exhibit strong general image understanding in WebQA, they struggle with complex reasoning, with most scoring below 50% and some as low as 2% (e.g., Fuyu-8B in English), alongside notable multilingual performance drops (Figure 2). In Web Code Editing, top-performing models like Gemini-2.0-Flash and Claude-3.5-Sonnet outperform open-source counterparts, yet still struggle with maintaining logical structure and syntactic correctness, highlighting the need for more structure-aware code-editing techniques, particularly for complex modifications. Similarly, in Mockup2Code, models such as OpenAI-o1 and Claude-3.5 achieve a high LLM-as-Judge score (4/5) on simple layouts but fail with nested element structures, revealing limitations in UI hierarchy comprehension. These findings emphasize the need for improved multimodal alignment, UI-aware modeling, and cross-lingual robustness to bridge the gap between vision-language models and real-world web interaction.

Our contributions are as follows:

- **Comprehensive Multi-Task Benchmark**: A unified evaluation suite encompassing website VQA, web design-to-code generation, and code editing tasks.
- **Diverse, Expert-Annotated Multilingual Data**: Fine-grained question-answer pairs, code edits, and UI design annotations across four languages, enabling comprehensive evaluation.
- **Findings**: MLLMs face challenges in multi-step reasoning and grounding for WebQA, precise code editing, UI hierarchy understanding in web development, and multilingual generalization.

2 Related Work

Web Understanding and Agentic MLLMs. Multimodal learning has become central to web UI understanding, integrating visual, textual, and

structural modalities to support both web comprehension and agentic navigation. Early work, such as Screen2Words (Wu et al., 2021), parsed web screenshots into UI elements, later influencing MLLM pretraining(Lee et al., 2023). Recent advances leverage patching strategies(Baechler et al., 2024), grounding(Cheng et al., 2024), text-structural alignment(Xu et al., 2024; Bai et al., 2021), and context-aware UI representations(Kil et al., 2024). These innovations have expanded MLLM applications in web agents, enabling models to navigate and manipulate websites based on user instructions (Zheng et al.; Yoran et al., 2024; Cheng et al., 2024). However, existing benchmarks often rely on limited artificial websites(Deng et al., 2024; Zhou et al., 2023) or focus solely on English data(He et al., 2024; Lù et al., 2024; Zhang et al., 2024; Chen et al., 2024a), lacking diversity and real-world complexity. WebMMU addresses these gaps by incorporating real-world websites and multilingual queries, requiring models to perform complex reasoning and UI grounding, making it a more comprehensive evaluation framework for MLLM-driven web understanding and navigation.

Visual Question Answering for Web. Progress in web-based VQA has been driven by benchmarks like WebSRC (Chen et al., 2021b), WebQA (Chang et al., 2022), WebQuest (Wang et al., 2024a), VisualWebBench (Liu et al., 2024), and WebWalkerQA (Wu et al., 2025) covering tasks such as captioning, webpage QA, and element grounding. Compared to traditional VQA on natural images (Yue et al., 2024), web-based VQA additionally requires understanding structured webpage layouts, the relationships between UI elements, and their functional roles within web environments. However, these benchmarks cover limited tasks, domains and languages. WebMMU addresses this gap by covering 20 domains in four languages and adding fine-grained categories like action grounding, multi-step reasoning, and general understanding for more thorough evaluation.

Automatic Web Design and Development. Code generation and editing have been widely studied across programming languages, with benchmarks evaluating code generation (Chen et al., 2021a; Jimenez et al., 2024; Rodriguez et al., 2024b,a) and code editing based on natural language instructions (Guo et al., 2024; Tian et al., 2024). However, most previous studies focus on general-purpose programming, neglecting web design and development. To bridge this gap, Gui et al. (2024) and Yun et al. (2024) explore generating HTML/CSS from web screenshots. In contrast, WebMMU introduces Web Code Editing, which involves multilingual tasks for modifying a website's visual and functional features based on user instructions, better reflecting real-world web development use cases. Additionally, WebMMU includes Mockup2Code; unlike prior work (Jain et al., 2019; Barua et al., 2022) that relies on simplistic and artificial sketches drawn by researchers, our sketches are extracted from real-world websites, preserving complex element hierarchies through expert annotation.

3 WebMMU Benchmark

In this section, we introduce WebMMU, detailing its data collection, annotation, and task overview.

3.1 Data Collection and Annotation

Website Selection and Data Capture. To construct WebMMU, we curated a diverse set of webpage URLs from the FineWeb dataset (Penedo et al., 2024) and applied domain-specific heuristics to ensure coverage across 20 popular, content-rich, and feature-rich web domains (e.g., shopping, booking, technology). We selected webpages in four languages: English, German, French, and Spanish – considering linguistic diversity, annotator availability, and budget constraints. To capture full browsing sessions on a single webpage, we generated collages combining multiple snapshots taken at different scroll depths and interaction states within the page. A viewport-specific snapshot was retained alongside relevant HTML and assets (e.g., CSS, JavaScript). Selection strictly adhered to web crawling policies (e.g., `robots.txt`).

Annotation Process. Annotators were provided with webpage screenshots, corresponding HTML, and asset files and were tasked with three objectives: (1) generating open-ended and multiple-choice questions that capture real-world usage, including highlighting, clicking, and multi-step reasoning; (2) creating UI mockups of varying complexity and formats to support design-to-code workflows; and (3) formulating code edit requests that require programming expertise. A structured training phase ensured annotation consistency and quality. Further details on annotator guidelines are given in the Appendix A.

Quality Control and Annotator Demographics. A 100% quality assurance framework was implemented in three stages: *Trainer Review*, where experienced annotators performed initial annotations; *Primary QA (QA1)*, where independent specialists verified accuracy, completeness, and adherence to guidelines; and *Secondary QA (QA2)*, ensuring consistency with expert-level annotation criteria. The dataset was annotated by 127 professionals across North America, South America, Europe, Africa, and Asia, representing diverse linguistic and domain expertise. English annotators primarily came from Asia, German and French from Europe, and Spanish from Latin America. Annotators held qualifications ranging from bachelor's to advanced degrees for specialized tasks and were compensated above fair market wages, ensuring ethical labor practices and high-quality results.

3.2 Tasks Overview

3.2.1 Web Question Answering (WebQA)

The WebQA task in WebMMU evaluates models' ability to extract, integrate, and ground structured UI elements, numerical data, and graphical components from web screenshots while reasoning over hierarchical layouts, predicting actions, and ensuring spatial grounding. It consists of three categories: **Agenctic Action**, which focuses on web navigation and action execution without feedback from the environment, requiring models to understand UI elements like buttons, menus, and hyperlinks, identify elements (e.g., *"Where can I find the coaching plans?"*), and execute actions (e.g., *"How can I save this drill?"*) while handling *spatial grounding* and distinguishing *static vs. interactive elements* across multilingual UIs; many of these tasks also require *coordinate-based reasoning* to localize UI components accurately. **Multi-step Reasoning** involves multi-step inference, numerical calculations, and comparisons across UI components (e.g., *"If a customer were to buy all the camera models mentioned on the bottom of this page in Ëxpert Camera Reviewsïable, what would be the grand total?"*), requiring models to integrate text, numerical values, and layout structures from structured web content, where hierarchical reasoning is essential despite being constrained to single-frame snapshots; and **General Visual Comprehension**, which assesses a model's ability to extract and synthesize structured and unstructured data from web screenshots, including OCR-extracted text, images, graphical elements, and UI components (e.g., *"How many brand logos are in the Featured Brands section?"*), emphasizing *semantic comprehension* beyond standard OCR-based extraction.

While existing web VQA benchmarks such as WebSRC (Chen et al., 2021b), WebQA (Chang et al., 2022), and VisualWebBench (Liu et al., 2024) provide useful tasks for information extraction or element grounding, they often focus on isolated subtasks, rely on static screenshots or HTML-only views, and primarily target English content. In contrast, WebMMU integrates action understanding, spatial grounding, and multi-step reasoning across four languages, making it a more comprehensive testbed for evaluating multilingual and agentic capabilities of MLLMs in real-world web environments. Unlike prior benchmarks, WebMMU includes questions with coordinate-based UI element localization, logical reasoning across DOM hierarchies, and multilingual UI semantics; all within a single benchmark and on realistic, domain-diverse web data.

3.2.2 Mockup2Code

The Mockup2Code task in WebMMU advances design-to-code by translating hand-drawn wireframes and high-fidelity digital mockups into structured code. Unlike text-based UI generation, it evaluates a model's ability to interpret spatial hierarchies and UI structures from visual inputs. The dataset includes low-fidelity sketches and digitally created mockups, challenging models to generalize across abstraction levels in web design while tackling component recognition, spatial alignment, and structured code synthesis. Unlike prior design-to-code datasets, WebMMU incorporates real-world web layouts, ensuring models generate syntactically correct and semantically meaningful code aligned with modern web development practices. Prior design-to-code efforts like Pix2Code (Beltramelli, 2018), Sketch2Code (Jain et al., 2019), and WebSight (Laurençon et al., 2024) typically rely on synthetic or simplified UI inputs, often in English, and lack the layout depth and language variation present in real-world designs. WebMMU advances this space by incorporating expert-verified mockups drawn from real websites, supporting multilingual content, and emphasizing nested UI hierarchies. While prior datasets assess UI generation in isolation, our task evaluates models' ability to handle real, noisy design inputs and produce web-standard HTML/CSS outputs under

spatial and structural constraints, providing a more realistic and multilingual benchmark for layout-to-code modeling.

3.2.3 Web Code Editing

Web Code Editing is a novel task, which evaluates a model's ability to modify webpage code while preserving functional and structural integrity, given a screenshot, source code, and a user edit request. To perform well, models must complete three sub-tasks: (1) understand the provided inputs, including the webpage codebase, visual elements in the screenshot, and the requested modification; (2) identify the relevant code snippets that require modification; and (3) generate the appropriate HTML, CSS, or JavaScript edits to implement the requested change. These sub-tasks require an advanced understanding of webpage development and realistic code editing capabilities. The modification requests span a broad range of visual and functional changes. Visual edits include adjusting font size and colors, repositioning elements, and adding headers or footers. Functional modifications involve adding interactive components such as buttons or forms and enhancing user experience with dynamic UI elements. The task is multilingual, aligning with the broader scope of WebMMU. Given the length of webpage source code, models are prompted to output only the necessary code differences rather than rewriting the entire codebase. This improves both practicality and efficiency, ensuring that the generated edits remain concise and targeted. More details on the prompt formulation are provided in Appendix C.2. Existing benchmarks such as CodeEditorBench (Guo et al., 2024), InstructCoder (Li et al., 2023), and CanItEdit (Cassano et al., 2023) evaluate general-purpose code editing in languages like Python or JavaScript, often lacking visual context or domain specificity. In contrast, WebMMU focuses on precise modifications to HTML/CSS/JS code grounded in user-facing web interfaces. It uniquely incorporates multilingual edit instructions and aligns edits with visual page cues, simulating realistic web development workflows. Unlike benchmarks driven purely by unit test correctness, WebMMU emphasizes functional and structural preservation within existing codebases, thereby offering a more targeted and practical benchmark for UI-aware and multilingual web code editing.

	En	Es	De	Fr	Total
Website Images	834	418	416	391	**2059**
WebQA	1976	1521	1201	1404	**6102**
Mockup2Code	180	93	85	78	**436**
Web Code Editing	886	252	226	238	**1602**

Table 1: **Dataset Statistics.** Language-wise dataset breakdown across tasks. We report the number of web images per language. English (En), Spanish (Es), German (De) and French (Fr).

Task	Metric	Evaluation Details
WebQA	LLM-as-Judge	Accuracy; 0 (incorrect) / 1 (correct).
Mockup2Code	LLM-as-Judge	Layout fidelity on a 1-5 scale (layout, spacing, grid).
Code Editing	BLEU, Tree-BLEU	Design and structural correctness.
	LLM-as-Judge	Functional correctness on a 1-5 scale.

Table 2: **Evaluation Metrics** used in WebMMU.

3.3 Dataset Statistics

WebMMU covers four languages: English, Spanish, German, and French (see Table 1). It contains 2059 webpage images from domains like e-commerce, education, news, and finance. It includes 6102 WebQA samples, 436 Mockup2Code instances, and 1602 Web Code Editing cases. Unlike previous datasets that focus on predefined UI layouts, WebMMU uses full-page snapshots, including dynamic content, nested structures, and multimodal dependencies. A small portion of images consist of multiple panels combined into a single image, capturing dense information and replicating browsing sessions.

4 Evaluation

We evaluate state-of-the-art MLLMs across both closed-source and open-source categories. Model inference for WebQA, Mockup2Code, and Web Code Editing follows standardized prompts (Appendix D). Evaluation combines LLM-as-Judge (Zheng et al., 2023) scoring with established automatic metrics, as summarized in Table 2.

LLM-as-Judge is used to evaluate WebQA, where model responses receive binary correctness scores (0 or 1) based on predefined criteria for semantic accuracy and reasoning completeness (Appendix D.3). This structured approach ensures consistency and prevents arbitrary grading. Inspired by automated evaluation in image synthesis (Ku et al., 2023), Mockup2Code uses LLM-as-Judge,

Model	English			French			German			Spanish		
Claude3.5 Sonnet	48.1	2.9	47.1	48.2	**14.6**	46.3	34.7	**16.0**	41.0	62.4	16.1	49.3
Gemini-2.0-Flash	44.3	1.6	47.6	41.4	10.8	42.1	29.8	12.1	41.7	51.1	11.1	44.2
OpenAI-o1	**68.2**	**4.9**	**72.7**	**55.5**	12.3	**69.6**	**46.4**	14.9	**49.6**	**66.0**	15.3	**60.9**
Phi3.5-VI-4b (Abdin et al., 2024)	8.0	0.8	15.7	2.0	6.2	21.6	6.9	10.9	21.4	9.9	4.1	23.7
UI-Tars-7b (Qin et al., 2025)	17.3	0.5	42.3	9.4	4.1	38.0	9.6	5.6	38.5	11.9	2.3	33.2
Molmo-7b (Deitke et al., 2024)	16.5	0.0	40.6	6.4	4.2	**49.6**	5.1	9.6	29.1	6.5	4.3	27.7
Qwen2VL-7B (Wang et al., 2024b)	19.3	1.3	43.4	14.9	8.5	37.7	19.4	11.0	36.4	17.2	8.4	38.9
Llava-OV-7B (Li et al., 2024a)	18.9	2.3	**39.9**	8.7	8.3	36.3	12.0	10.6	35.6	7.5	6.8	33.2
Qwen2.5VL 7B (Bai et al., 2025)	**35.2**	**30.8**	**51.4**	**27.2**	**25.4**	41.6	**23.6**	**22.7**	**40.6**	**32.6**	**20.5**	**44.8**
Fuyu-8b (Bavishi et al., 2023)	0.7	0.3	5.7	0.0	0.6	5.1	0.2	0.6	2.5	0.2	0.2	3.3
Internvl2.5-8b (Chen et al., 2024b)	23.3	1.4	36.2	13.1	6.0	30.5	15.7	10.9	34.8	14.2	6.2	33.8
Pixtral-12b (Agrawal et al., 2024)	24.5	**2.9**	38.5	**19.2**	**10.1**	**53.5**	12.8	**14.2**	26.4	**21.6**	**13.7**	32.3
Llama-3.2-11B-Vision (Dubey et al., 2024)	**29.9**	1.8	38.8	11.0	8.0	33.3	**17.9**	12.2	**36.6**	20.3	5.8	**34.7**
Llava-OV-72B (Li et al., 2024a)	31.7	1.9	40.2	16.4	8.2	39.0	23.6	11.7	42.4	23.3	7.4	39.2
Molmo 72B (Deitke et al., 2024)	25.8	1.1	40.1	18.5	7.1	40.4	23.3	12.0	36.1	26.0	7.5	38.6
Qwen2VL-72B (Wang et al., 2024b)	33.7	1.0	44.2	28.1	11.6	42.0	28.2	11.8	42.9	37.5	10.3	43.6
Internvl2.5-38b (Chen et al., 2024b)	33.6	1.4	45.0	30.2	9.9	47.9	28.1	13.4	42.5	36.7	11.8	45.7
Qwen2.5VL 72B (Bai et al., 2025)	**43.4**	**37.1**	**52.8**	**36.8**	**36.5**	**47.9**	**31.2**	**28.7**	**44.5**	**44.9**	**41.4**	**49.5**

Table 3: **WebQA Performance.** Model accuracy (%) by question type and language. : Multi-step Reasoning, : Agenctic Action, : General Visual Comprehension. Best models per size category are in **bold**. Model sizes: blue (<8B params), orange (8–12B), green (>12B), gray proprietary.

assessing the alignment between input sketches and rendered outputs across three key dimensions: *layout structure, spacing, and grid consistency* (Appendix D.2). Each aspect follows well-defined scoring guidelines, ensuring reproducible and fair assessments. For Web Code Editing, we evaluate both *structural correctness* and *functional accuracy*. The former is measured using BLEU (Papineni et al., 2002) and TreeBLEU (Gui et al., 2024), ensuring syntactic validity and adherence to coding conventions. The latter relies on LLM-as-Judge, where functional equivalence between reference and predicted edits is rated on a 1-5 scale. To avoid arbitrary scoring, rating criteria explicitly define correctness levels based on functional preservation and intended user modifications. Since web functionalities can be implemented in multiple ways, the evaluation accounts for semantically valid alternatives, preventing undue penalization of syntactically different but functionally correct edits. For all LLM-as-Judge evaluations, we use GPT4o-1120, which has demonstrated strong alignment with human judgment and diverse scoring behavior (Feizi et al., 2025), ensuring robustness across tasks.

5 Results

5.1 WebQA Performance

Table 3 presents model accuracy for three question types. Closed-source models, such as Gemini 2.0 Flash and Claude 3.5 Sonnet, outperform open-source alternatives across all tasks but still struggle with agentic action, particularly in predicting spatial coordinates for interactive elements. Among open-source models, larger architectures (>30B parameters) like Qwen2.5VL-72B and Internvl2.5-38B perform better in general image understanding and UI recognition, while smaller models (<8B) exhibit poor generalization across tasks.

Performance varies by question type. General image understanding is easiest, relying mainly on visual recognition. Complex reasoning is harder, with most models scoring below 50% and some as low as 2% (e.g., Fuyu-8b in English), showing difficulties in retrieving and reasoning over structured webpage content. **Agentic action is the hardest, with top models rarely surpassing 10% accuracy**, as it requires precise spatial grounding, such as recognizing interactive elements (e.g., "About Me" in a menu) and predicting **approximate bounding boxes**. While many models detect interactive parts, they struggle with localization,

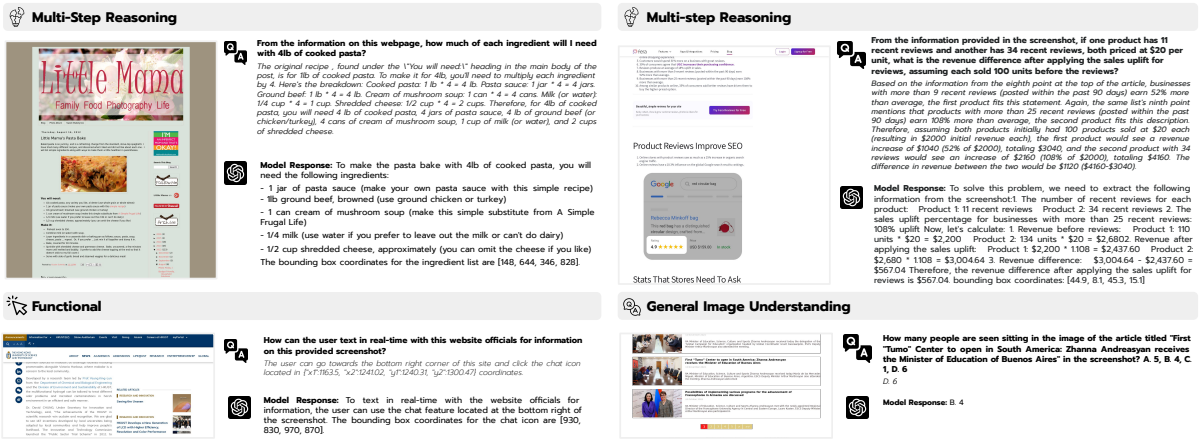


Figure 2: **Failure Cases in WebQA** for one of the top-performing open-source model (InternVL-38B). Most prominent errors occur in grounding actions (e.g., identifying clickable link coordinates) and multi-step reasoning tasks, such as detailed step-by-step calculations.

resulting in low scores.

Error Analysis. Figure 2 reveals common failures: models often **miscalculate numbers or fail in multi-step reasoning**. In agentic action, inaccurate bounding boxes hurt performance. Multilingual generalization also remains weak despite resource-rich languages. These issues highlight the need for better spatial reasoning, numerical understanding, and cross-lingual adaptation to close the gap between vision-language models and real web interaction.

5.2 Mockup2Code Generation

Figure 3 evaluates the Mockup2Code task, reporting scores for each dimension and overall performance. Open MLLMs such as Phi3.5-VI, Fuyu-8B, and GLM4V-9B generally perform poorly across all metrics. Notably, Phi3.5-VI and Fuyu-8B score nearly 1 across all dimensions, indicating a complete failure on this task. Nevertheless, performance improves with model scale. For instance, Qwen2VL's score rises from 1.90 to 3.39 when scaling from 7B to 72B, while InternVL2.5 improves from 2.34 to 3.61 when scaling from 8B to 38B. Additionally, Pixtral-12B outperforms all 7B/8B models. Still, even the best open MLLMs struggle, especially with complex designs – InternVL2.5-38B, the highest performer, scores only 2.98 out of 5. In contrast, proprietary models like Claude-3.5, Gemini-2.0-Flash, and OpenAI-o1 perform significantly better, particularly on simple UI designs, where they achieve LLM-as-Judge scores above 4. **However, their performance declines in complex variants, with top scores reaching only 3.4 out of 5.** Across all evaluation dimensions, both proprietary and large-scale open MLLMs struggle most with spacing, which requires accurately setting element dimensions and margins based on sketch input.

Case Analysis. Figure 5 shows both success and failure cases of the top model OpenAI-o1 on Mockup2Code. OpenAI-o1 handles simple, flat layouts well, even with moderate element variety and count. However, it struggles with nested structures, often misaligning and failing to preserve element hierarchy and spacing, especially when <p> and <a> tags are nested within . Similar issues occur in other samples and models, as seen in Figures 14 and 16.

5.3 Code Editing Performance

Figure 4 shows Web Code Editing results evaluated by LLM-as-Judge (metrics in Table 7). Proprietary models achieve the highest functional accuracy, but only marginally outperform large open-source models, **indicating both struggle to preserve functional correctness alongside syntactic consistency**. Smaller models like Phi3.5-VI and Fuyu-8b perform worst, often failing to generate valid code (score <1.5). Performance improves with size; Qwen2VL-72B and InternVL2.5-38B rival closed-source models. Yet, even the strongest exhibit clear limitations producing structurally correct edits that fully preserve functionality. Multilingual performance is stable for top models but more variable for smaller ones, reflecting challenges in adapting edits across languages. Crucially, all models – especially open-source – fail to **automatically**

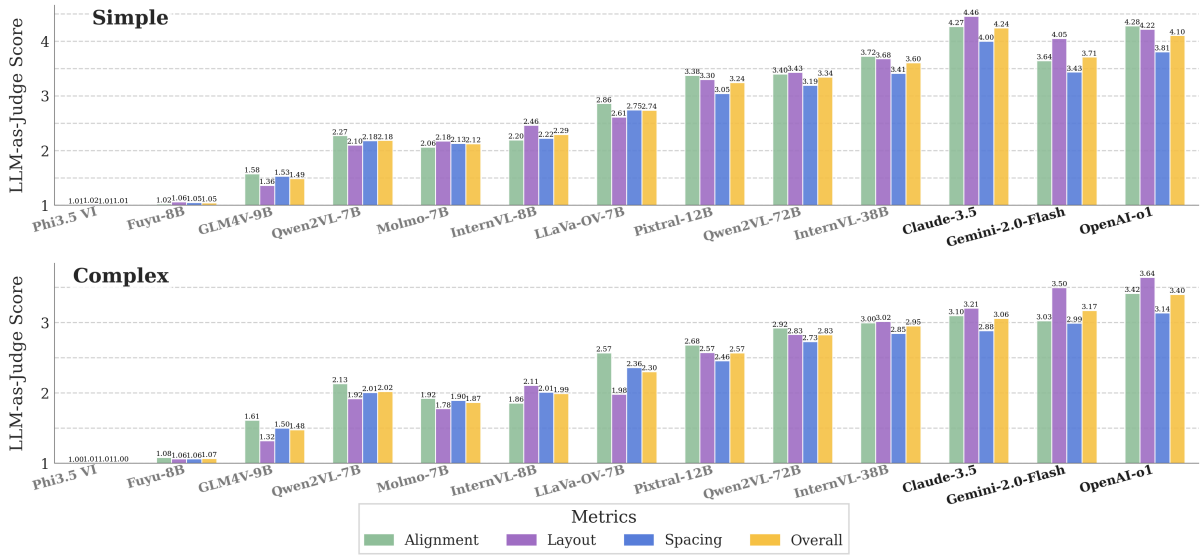


Figure 3: **Mockup2Code Performance.** LLM-as-Judge evaluation scores for simple and complex UI mockups across three key dimensions: *alignment, layout, and spacing*, along with overall performance. Higher scores indicate better fidelity between the generated and reference web designs. Closed-source models outperform open-source alternatives, particularly in complex cases, yet challenges remain in high-fidelity code generation.

generate valid patch files for seamless source integration. Despite access to full source files, none produced patch content directly usable without manual fixes, making human oversight essential and highlighting a core challenge in automating web code edits.

5.4 Metric-Human Alignment

We sampled 100 examples per task and enlisted PhD students and researchers as annotators to evaluate alignment between human judgments and the automatic metric (LLM judge). For WebQA, humans agreed with the LLM judge in 89% of cases. Most disagreements involved functional questions, where the judge required exact bounding boxes, but humans were more lenient – accepting answers that correctly identified the clickable link location without a precise bounding box (e.g., the "about us" link in the navbar). We consider the judge's stricter criteria correct since models were prompted to provide exact bounding boxes for such questions (see Appendix C.1). For Mockup2Code Spearman correlations were 0.39 (layout), 0.33 (spacing), and 0.46 (alignment), averaging 0.43 overall. Pearson correlations were slightly higher: 0.42 (layout), 0.41 (spacing), 0.48 (alignment), with an overall average of 0.50. These correlations, while moderate, reflect the task's subjectivity and support the reliability of the automatic evaluation. For Web Code Editing, expert annotators validated the LLM judge's assessments with 91% accuracy, demonstrating both the reliability of the evaluation and the validity of the associated judge's rationales assigned.

6 Conclusion

WebMMU evaluates MLLMs on a real-world, challenging web question answering task and two code generation tasks: front-end design and code editing. Our tasks cover four languages and a wide variety of domains, sourced from human annotators. Our results show that Web VQA models struggle with interpreting complex UIs, reasoning, and multilingual generalization. Code editing models often generate syntactically correct but logically inconsistent code. UI generation models face a trade-off between precise element placement and preserving the original design's meaning. These challenges underscore the need for enhanced multimodal alignment, UI-aware architectures, and robust cross-lingual adaptation to develop future web agents capable of effectively performing a wide range of human tasks on the web.

Limitations While WebMMU provides a comprehensive evaluation of web-based AI reasoning and code generation, it has several limitations. First, it is restricted to single-screenshot web reasoning, capturing static snapshots rather than supporting interactive environments or multi-turn nav-

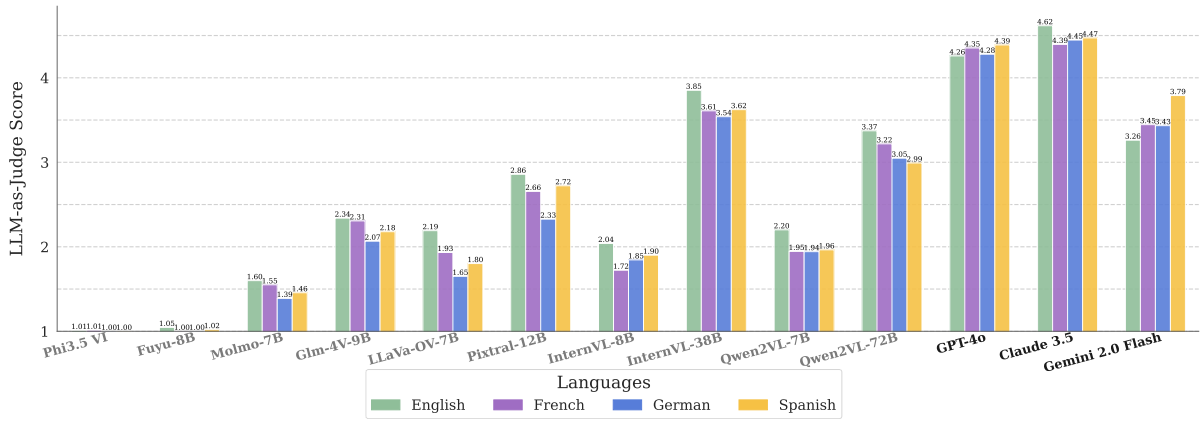


Figure 4: **Performance on Code Edits.** LLM-as-Judge metric, on a scale of 1-5, used to evaluate functional correctness of code edits. All models, including closed-source models, struggle with the Web Code Editing task of WebMMU. Refer to Table 7 for full results, including BLEU and TreeBLEU scores, of all models.

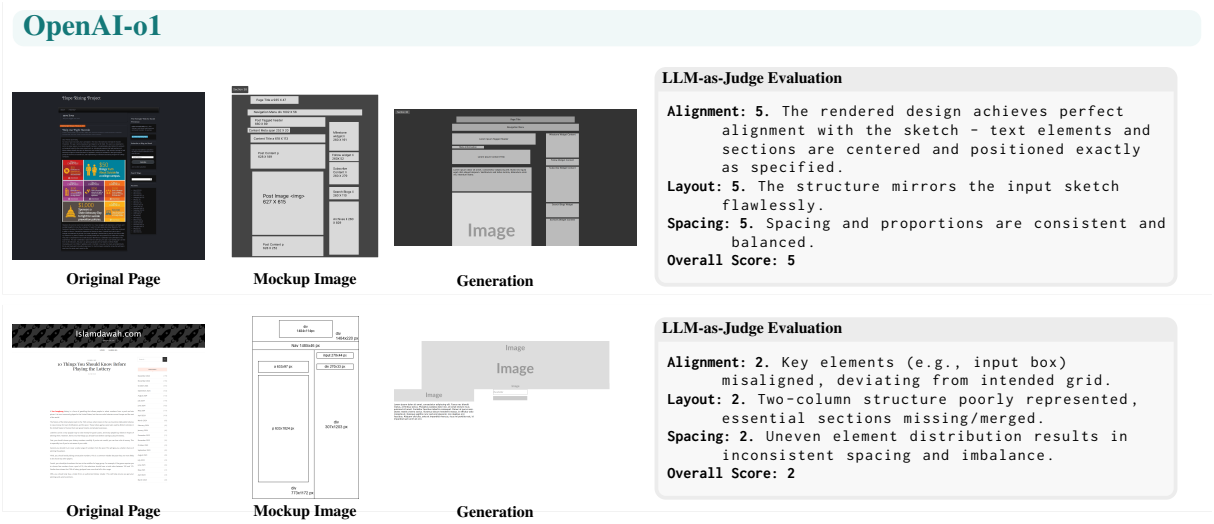


Figure 5: Success (top) and failure (bottom) cases for Mockup2Code Generation from OpenAI-o1.

igation. Although multi-step reasoning tasks are included, they rely solely on single-image (including multiple panels of a browsing session), limiting evaluation in dynamic web exploration. Second, linguistic coverage is constrained to four languages: English, French, German, and Spanish; due to annotator availability, which may limit generalization to underrepresented languages and regional web structures. Third, while Mockup2Code and Web Code Editing cover core web technologies such as HTML, CSS, and JavaScript, modern frontend frameworks like React, Angular, and Vue.js are not explicitly evaluated. Finally, the automatic LLM judge metric, though reliable and fast, does not fully replicate human evaluation. Future work could explore improved automatic metrics or hybrid evaluation approaches to better capture nuanced human judgments.

Ethical Considerations WebMMU is a benchmarking resource designed strictly for research purposes in multimodal and multilingual web understanding and generation. All tasks are created by human annotators using everyday web content and undergo thorough validation, so we do not anticipate misuse or harmful content. Compared to prior work, WebMMU expands evaluation across multiple languages, though coverage remains limited by annotator availability. To the best of our knowledge, the dataset contains no NSFW or harmful content. We commit to promptly removing any data upon valid requests once publicly released.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat

Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, Soham Ghosh, Amélie Héliou, Paul Jacob, Albert Q. Jiang, Kartik Khandelwal, Timothée Lacroix, Guillaume Lample, Diego Las Casas, Thibaut Lavril, and 23 others. 2024. Pixtral 12b. *Preprint*, arXiv:2410.07073.

AI Anthropic. 2024. Claude 3.5 sonnet model card addendum. *Claude-3.5 Model Card*, 3(6).

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and 1 others. 2021. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Somoy Subandhu Barua, Imam Mohammad Zulkarnain, Abhishek Roy, Md Golam Rabiul Alam, and Md Zia Uddin. 2022. Sketch2fullstack: Generating skeleton code of full stack website and application from sketch using deep learning and computer vision. *arXiv preprint arXiv:2211.14607*.

Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. 2023. Introducing our multimodal models.

Tony Beltramelli. 2018. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI symposium on engineering interactive computing systems*, pages 1–6.

Federico Cassano, Luisa Li, Akul Sethi, Noah Shinn, Abby Brennan-Jones, Jacob Ginesin, Edward Berman, George Chakhnashvili, Anton Lozhkov, Carolyn Jane Anderson, and 1 others. 2023. Can it edit? evaluating the ability of large language models to follow code editing instructions. *arXiv preprint arXiv:2312.12450*.

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. 2022. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16495–16504.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, and 1 others. 2024a. Gui-world: A dataset for gui-oriented multimodal llm-based agents. *arXiv preprint arXiv:2406.10819*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021a. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xingyu Chen, Zihan Zhao, Lu Chen, Jiabao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021b. Websrc: A dataset for web-based structural reading comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4173–4185.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, and 1 others. 2024b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, and 32 others. 2024. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aarash Feizi, Sai Rajeswar, Adriana Romero-Soriano, Reihaneh Rabbany, Spandana Gella, Valentina Zantedeschi, and João Monteiro. 2025. Pairbench: A systematic framework for selecting reliable judge vlms. *arXiv preprint arXiv:2502.15210*.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Yi Su, Shaoling Dong, Xing Zhou, and Wenbin Jiang. 2024. Vision2ui: A real-world dataset with layout for code generation from ui designs. *arXiv preprint arXiv:2404.06369*.

Jiawei Guo, Ziming Li, Xueling Liu, Kaijing Ma, Tianyu Zheng, Zhouliang Yu, Ding Pan, Yizhi Li, Ruibo Liu, Yue Wang, and 1 others. 2024. Codeeditorbench: Evaluating code editing capability of large language models. *arXiv preprint arXiv:2404.03543*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Vanita Jain, Piyush Agrawal, Subham Banga, Rishabh Kapoor, and Shashwat Gulyani. 2019. Sketch2code: transformation of sketches to ui in real-time using deep neural network. *arXiv preprint arXiv:1910.08930*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *ICLR*.

Jihyung Kil, Chan Hee Song, Boyuan Zheng, Xiang Deng, Yu Su, and Wei-Lun Chao. 2024. Dual-view visual contextualization for web navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14445–14454.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. 2023. Viescore: Towards explainable metrics for conditional image synthesis evaluation. *arXiv preprint arXiv:2312.14867*.

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*.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Kaixin Li, Qisheng Hu, Xu Zhao, Hui Chen, Yuxi Xie, Tiedong Liu, Qizhe Xie, and Junxian He. 2023. Instructcoder: Instruction tuning large language models for code editing. *arXiv preprint arXiv:2310.20329*.

Ryan Li, Yanzhe Zhang, and Diyi Yang. 2024b. Sketch2code: Evaluating vision-language models for interactive web design prototyping. *arXiv preprint arXiv:2410.16232*.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

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

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Juan Rodriguez, Xiangru Jian, Siba Smarak Panigrahi, Tianyu Zhang, Aarash Feizi, Abhay Puri, Akshay Kalkunte, François Savard, Ahmed Masry, Shravan Nayak, Rabiul Awal, Mahsa Massoud, Amirhossein Abaskohi, Zichao Li, Suyuchen Wang, Pierre-André Noël, Mats Leon Richter, Saverio Vadacchino, Shubbam Agarwal, and 24 others. 2024a. Bigdocs: An open and permissively-licensed dataset for training multimodal models on document and code tasks. *arXiv preprint arXiv:2412.04626*.

Juan A. Rodriguez, Abhay Puri, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. 2024b. Starvector: Generating scalable vector graphics code from images and text. *Preprint*, arXiv:2312.11556.

Runchu Tian, Yining Ye, Yujia Qin, Xin Cong, Yankai Lin, Yinxu Pan, Yesai Wu, Haotian Hui, Weichuan Liu, Zhiyuan Liu, and 1 others. 2024. Debugbench: Evaluating debugging capability of large language models. *arXiv preprint arXiv:2401.04621*.

Maria Wang, Srinivas Sunkara, Gilles Baechler, Jason Lin, Yun Zhu, Fedir Zubach, Lei Shu, and Jindong

Chen. 2024a. Webquest: A benchmark for multimodal qa on web page sequences. *arXiv preprint arXiv:2409.13711*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024b. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024c. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jason Wu, Xiaoyi Zhang, Jeff Nichols, and Jeffrey P Bigham. 2021. Screen parsing: Towards reverse engineering of ui models from screenshots. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 470–483.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*.

Linhui Xu, Yawen Zhang, Yawen Li, Yawen Zhang, and Yawen Li. 2024. Hierarchical multimodal fine-grained modulation for visual grounding. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1–9. ACM.

Ori Yoran, Samuel Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. 2024. Assistantbench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8938–8968.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, and 1 others. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, and 1 others. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. *arXiv preprint arXiv:2406.20098*.

Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. 2024. Mmina: Benchmarking multihop multimodal internet agents. *arXiv preprint arXiv:2404.09992*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Human Annotator Instruction

A.1 WebQA Annotations Guideline

You will be provided with screenshots of websites. Your task is to create challenging questions that test deep understanding and reasoning about the image content. Each question should fall into one of the three categories described below, and be designed to encourage a detailed analysis of the screenshot. **Important Note:** If a screenshot lacks sufficient content or context for creating questions in any of the categories, mark the image as "Not enough content" and move to the next.

Agenctic Action Purpose: Focus on the interactive elements and navigation aspects of the website. These questions should prompt the viewer to interpret or locate specific functional elements, like buttons, menus, or links, and understand their purpose. Example: "Where would a user click to access their saved items?" Guidelines: Create questions that require the viewer to understand how different elements work or what actions they might trigger. Avoid overly simple questions that don't involve interaction or navigation. Do provide the bounding box location or hint on how to navigate.

Multi-step Reasoning Purpose: These questions should require multi-step thinking, involving the analysis of multiple parts of the image, comparisons, or drawing inferences from the content. Example: "How does the timing of updates in different news sources on this page provide insights into the event's coverage?" Guidelines: Formulate questions that connect elements across the image or require interpretation of trends, relationships, or content hierarchy. These should not be answerable from a single part of the image. If answerable, then

should be difficult e.g. solving a math question (see example) or asking what will happen if the cart is doubled (see example).

General Visual Comprehension Purpose: Assess the viewer's ability to identify and comprehend basic information displayed in the image, such as titles, labels, or the overall structure. Example: "What is the main title or header of this page?" Guidelines: Keep these questions straightforward, focusing on textual or visual elements that convey the primary purpose or information displayed. Aim for questions that require attention to specific details rather than general impressions. Highlight the region of answer with bounding box if needed (upto your choice).

A.2 Performing Code Editing on Websites

Understanding the Scope of Edits Before starting, identify the specific task or issue with clarity and precision. Ensure you fully understand the requested visual or functional changes before proceeding. Examples of tasks by difficulty are outlined below

Basic Changes

- Change the button color from blue to green.
- Fix a typo in the homepage headline.
- Remove the underlined style from all hyperlinks.
- Add a border to images in the gallery section.

Intermediate Enhancements

- Replace the navigation bar font with 'Roboto' and ensure it matches the design mockup.
- Add a hover effect to all buttons, changing their background to light gray.
- Update the footer links to open in a new tab and add appropriate ARIA labels for accessibility.
- Create a consistent color scheme for all headings on the page.

Advanced Functional or Design Tasks

- Add a new section to the homepage to showcase recent blog posts, styled to match the website theme.
- Refactor the JavaScript for the carousel to improve performance and fix the sliding bug.
- Optimize the CSS for faster page load times by combining redundant rules and removing unused classes.
- Implement a lightbox feature for viewing images in the gallery.
- Create a visually engaging header with a full-width background image and overlay text for the homepage.
- Design a custom 404 error page with an animated illustration and a link back to the homepage.
- Develop a visually interactive pricing table with hover effects to highlight selected options.
- Redesign the "About Us" section using a card layout for team member profiles, including images and bios.
- Update the contact form with a modern design, including floating labels and inline validation.
- Animate the scrolling experience for anchor links to smoothly transition between sections of the page.

Key Principles: a) Focus on Instructions. b) Only address the requested tasks and avoid unrelated changes unless explicitly instructed. c) Document Changes Clearly and d) For every modification, provide a clear record that includes:

- What was changed?
- Why was it changed?
- The location of the change (e.g., file name and line numbers, or element location in the inline HTML).

A.3 Performing Sketch Task

The distinction between simpler and more complex sketches typically depends on the number of components and the level of detail in the specifications. Simpler sketches usually have fewer elements (e.g., basic shapes, minimal labels), while complex sketches include multiple, interrelated components and detailed instructions (e.g., specifying dimensions, class names like 'div nav,' or

explicit layout details). To differentiate, consider: **Simple**: Basic wireframes or mockups with minimal annotations (e.g., a rectangle representing a button). **Complex**: Detailed designs specifying attributes (e.g., 'button 200px wide, div with class="nav"') or involving hierarchical or nested components.

B Task Samples

Tables 4, 5, and 6 present representative examples from the WebMMU dataset, covering WebQA, Mockup2Code, and Web Code Editing tasks. The WebQA task (Table 4) evaluates a model's ability to interact with webpage elements, recognize visual content, and perform complex reasoning based on structured UI components. The Mockup2Code task (Table 5) illustrates how webpage screenshots are converted into structured HTML representations, distinguishing between basic layout sketches and detailed UI component mappings. The Web Code Editing task (Table 6) demonstrates automated HTML modifications, providing before-and-after visual transformations based on functional and design-driven prompts. These task samples comprehensively showcase the challenges in webpage understanding, layout structuring, and automated UI refinement within the WebMMU benchmark.

C Model Output Generation Prompts

C.1 WebQA Task Completion Prompt

We present the prompt used for the WebQA task in Figure 6. The prompt instructs the model to analyze a website screenshot and provide a concrete answer to the given question. When the question requires identifying or interacting with specific elements on the screen, the model is asked to include the bounding box coordinates in its response.

Web QA Inference

Analyze the website screenshot and provide a detailed answer to the question. If the question involves locating or interacting with specific elements on the screen, include the bounding box coordinates [x_min, y_min, x_max, y_max] in your response.

Figure 6: Prompt for Generating Output of WebQA task

C.2 Web Code Editing Task Completion Prompt

This prompt guides a model in modifying the source code based on a modification instruction given by the user. The model outputs changes using the `git diff` format, highlighting additions and deletions with '+'s and '-'s respectively. This ensures clear and structured documentation of code edits. The prompt template can be seen in Figure 7.

C.3 Mockup2Code Task Completion Prompt

The Mockup2Code task involves generating HTML/CSS code from an input sketch (see Figure 8). Given a visual layout, the model must produce accurate, well-structured HTML and CSS that replicate the design. The prompt guides the model to interpret elements, hierarchy, and styling for faithful image-to-code conversion.

D Evaluation Prompts

This section provides details on the prompt formulations used throughout this work. These prompts guide the multimodal large language models in generating and evaluating responses across different tasks. The prompts are categorized based on their usage, including code modification, VQA evaluation, and UX scoring.

D.1 WebQA Evaluation Prompt

These prompts are used for evaluating model responses in VQA tasks. The model rates answers as 1 (Correct and Complete) or 0 (Incorrect or Irrelevant) based on factual accuracy and completeness. Example cases are provided to guide the evaluation. The prompt template can be seen in Figure 9.

D.2 Mockup2Code Evaluation Prompt

The Mockup2Code evaluation task involves assessing the accuracy of an MLLM-generated website based on an input sketch (see Figure 10). The evaluation prompt directs the annotator to compare the AI-generated HTML/CSS output with the given visual layout, ensuring that the generated website accurately replicates the design in terms of structure, styling, and layout. The evaluation criteria focus on layout structure, spacing, proportions, and alignment, allowing for a detailed assessment of how closely the generated output matches the intended design. The goal is to evaluate the model's

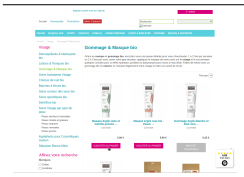
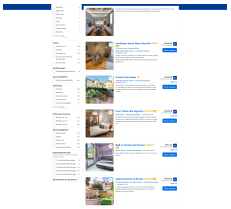
	How can I find more information about the player A.J. Dybantsa? Answer: Click on the "RIVALS150 ranking" at the lower left and select "A.J. Dybantsa" at (x1:230.34, x2:297.32, y1:1049.92, y2:1083.07).	How many players are visible in the 4th image on the left side? A) 2 B) 1 C) 4 D) 3 Answer: D) 3	Which user pays less when subscribing annually vs. monthly? Answer: - Yearly: $99.95 - Monthly: $9.95 × 12 = $119.40 - Savings: $19.45
	¿En qué área puede el usuario hacer clic para ver publicaciones anteriores? A) Categorías B) Lo más visto C) Archivo del blog Respuesta: C) Archivo del blog.	¿A qué tipo de público está dirigido este blog de recetas? Respuesta: Personas interesadas en repostería y cocina casera.	¿Cuál sería el efecto en la navegación si la sección "Lo más visto recientemente" estuviera al inicio? Respuesta: Aumentaría la accesibilidad de los artículos populares.
	Je cherche un soin pour le visage. Où dois-je aller ? Réponse: Dans la barre latérale sous "Soins visage par type de peau".	Quel est le prix total des articles si l'on exclut celui avec 5 étoiles ? A) 58,70 € B) 62,85 € C) 45,50 € D) 51,90 € Réponse: B) 62,85 €.	Avec un budget de 15 €, quels produits puis-je acheter ? Réponse: - Masque Argile verte et menthe poivrée bio (5,90 €) - Masque Argile rose bio - Peaux sensibles (5,90 €)
	Comment filtrer les hôtels qui acceptent les animaux ? Réponse: Dans la section "Ausstattung", cliquez sur "Haustiere erlaubt".	Combien de chaises sont visibles dans l'image de "Sunflower Santa Maria Novella" ? Réponse: 4 chaises.	Quelles sont les meilleures options d'hébergement près de Via Faenza ? Réponse: "Sunflower Santa Maria Novella" avec une note de 8.9.

Table 4: **WebMMU VQA Task Samples.** This table presents diverse Visual Question Answering (VQA) task samples from the WebMMU dataset, categorized into three types: (1) Functional (interaction with webpage elements), (2) General Understanding (visual recognition within webpage images), and (3) Complex Reasoning (logical inference and numerical computation). Each row showcases an input webpage image alongside representative questions and answers.

ability to interpret and transform the sketch into a functional, visually consistent website.

D.3 Code Edit Evaluation Prompt

This prompt is used to evaluate model responses in code edition tasks. The model rates answers as 1-5 (5 refers to the most correct and complete, and 1 refers to incorrect or irrelevant) based on factual accuracy and completeness. Example cases guide the evaluation. The prompt template can be seen in Figure 11.

E Case Studies of Model Performance

E.1 Case Studies for the Web Code Editing

We present case studies for the Web Code Editing task illustrating both success and failure examples. Figure 12 shows a success case where the Claude-3.5 model generates code that accurately follows the user's instructions. In contrast, Figure 13 highlights a failure case for the Gemini-2.0-Flash model, which overlooks key modifications requested by the user.

E.2 Case Studies for the Mockup2Code

We provide several examples illustrating the performance of different models on the Mockup2Code task, including both the input mockups and the generated outputs. Figure 14 shows failure cases where both the best closed-source model (OpenAI-o1) and the best open-source model (Internvl2.5-38b) struggled to accurately reproduce the designs. In contrast, Figure 15 highlights success cases for the OpenAI-o1 model, demonstrating its ability to handle both simple and complex mockups effectively. Additionally, Figures 16 and 17 present failure cases specifically for the open-source model Internvl2.5-8b and closed-source model OpenAI-o1, emphasizing areas where it underperforms on varying mockup complexities.

Web Code Editing Generation Prompt

You are an expert web developer specializing in identifying and applying modifications to web code. You will receive a website's screenshot and a combination of it's HTML, CSS, and/or JavaScript code, formatted as follows:

- **HTML Code:** `html_code`
- **CSS Code:** `css_code`
- **JavaScript Code:** `javascript_code`

You will also receive a modification prompt describing the required changes. Your task is to produce the necessary code modifications using 'git diff' format, even if some or all sections are missing. Follow these guidelines:

1. **Input code:** `<input_code>`
2. **Modification Prompt:** `<edit_prompt>`
3. **Output Diff:**
    - Use '+' for additions and '-' for deletions.
    - Modify only the relevant parts while preserving structure.
    - In case the code is missing, generate the necessary block of code from scratch.
    - Ensure readability and correctness in the modifications.

Only output the necessary diff; do not repeat the input code.

Figure 7: Web Code Editing generation prompt

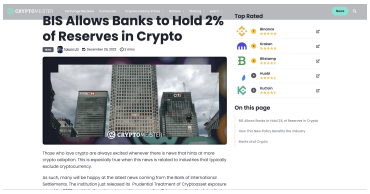
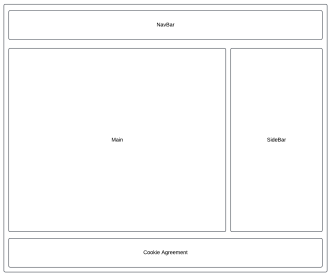
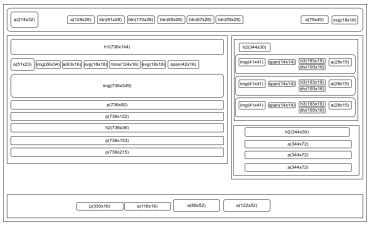
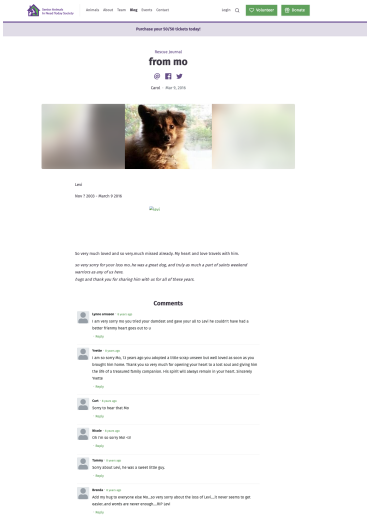
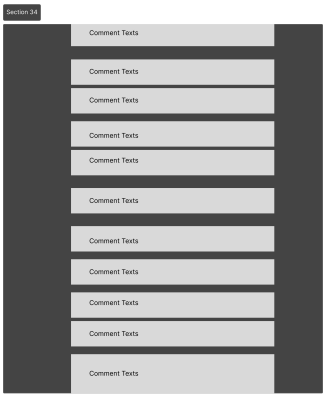
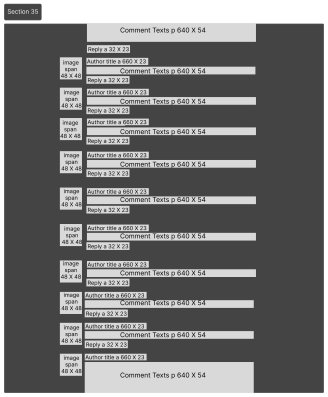
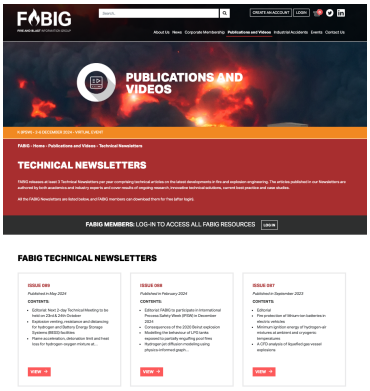
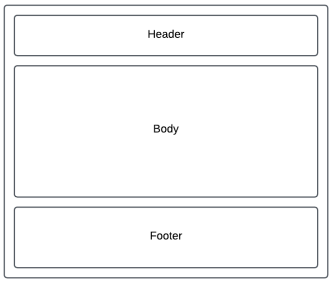
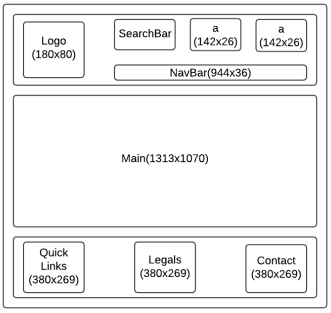
Input Image	Basic Layout Sketch	Detailed UI Representation
		
		
		

Table 5: **Mockup2Code Task Samples.** This table showcases examples from the Mockup2Code task, illustrating the transformation of webpage images into structured representations. Each row includes: (1) an Input Image (webpage screenshot), (2) a Simple Sketch (basic layout structure), and (3) a Complex Sketch (detailed UI components and text placements).

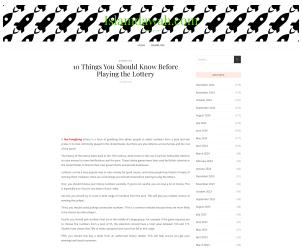
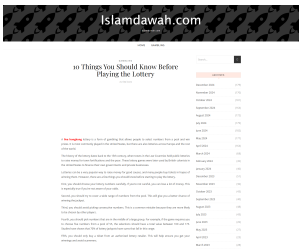
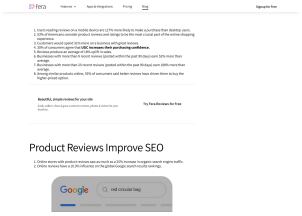
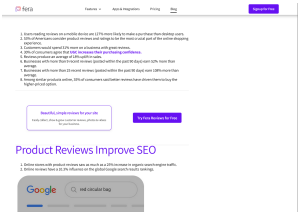
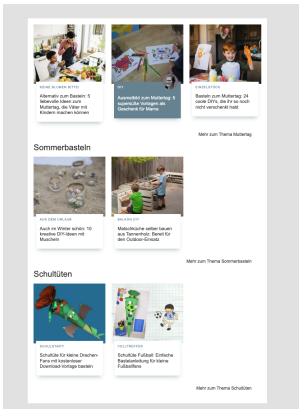
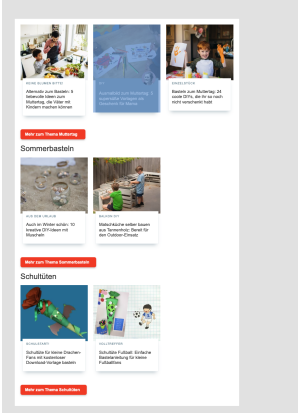
Input Image	Task Description	Rendered Image
	Comment faire pour afficher les différentes sections de l'article "La fête de la prune" en colonne et agrandir les images ?	
	Which changes should be made in the HTML code to improve the UI of the login form and navbar?	
	How can I fix the header element by adding a black overlay over the image, changing the font color to white, and setting the font family to "Lucida Sans"?	
	For a visually appealing design, enhance the navbar with hover and shadow effects, add hover interactions to buttons and links, and apply a card effect to containers.	
	Mache die drei Felder "link-next" auffälliger, indem du ihre Farbe, Größe oder Schriftstil anpasst.	

Table 6: **Web Code Editing Task Samples.** Code edition before and after screenshot of webpage.

Mockup2Code Generation Prompt

You are an expert website developer. Analyze the provided webpage sketch and generate a single, fully structured HTML file with embedded CSS that accurately reflects the design.
The output must be a self-contained HTML document with internal `<style>` tags for CSS. Ensure all elements are structured exactly as seen in the sketch—**no extra elements, no missing elements**.
HTML Requirements:

- **Components:** Include all necessary components such as headers, paragraphs, buttons, forms, and images, maintaining the correct hierarchy and placement.
- **Images:** Use images generated from `https://placehold.co/` with exact dimensions matching the sketch, a neutral background color, and centered "Image" text. For example: <img src="https://placehold.co/300x200?text=Image&bg=cccccc" alt="Placeholder Image">
- **Placeholder Text:** Use Lorem Ipsum for placeholder text where needed.

CSS Requirements:

- Implement CSS directly within the HTML file (inside a `<style>` block) to match the sketch, covering spacing, font sizes, colors, alignments, and element positioning.
- Use CSS Grid or Flexbox where appropriate to replicate the exact design layout.
- Apply styling for readability and interactive elements (e.g., fonts, button appearance).
- Ensure placeholder images maintain proper dimensions and design consistency.

Code Output:

- Provide a single, complete HTML file with internal CSS (do not separate them into different files).
- Do not include explanations, comments, or any extra formatting outside the code itself.

Figure 8: Mockup2Code Generation Prompt: It takes input sketch and outputs HTML/CSS code of the given input

Model	English			French			German			Spanish		
	BLEU	**TreeBLEU**	**LLM-as-Judge**	**BLEU**	**TreeBLEU**	**LLM-as-Judge**	**BLEU**	**TreeBLEU**	**LLM-as-Judge**	**BLEU**	**TreeBLEU**	**LLM-as-Judge**
QwenVL-7B	7.89	22.1	2.2	5.34	15.92	1.95	5.15	19.54	1.94	5.07	17.65	1.96
Molmo-7B	1.25	5.99	1.6	2.0	8.98	1.55	1.14	8.97	1.39	2.03	6.38	1.46
Phi-3.5-VI	0.01	0.0	1.01	0.0	0.0	1.01	0.03	0.0	1.0	0.03	0.0	1.0
Llava-OV-7B	7.78	20.34	2.19	3.66	13.6	1.93	5.14	19.26	1.65	3.54	14.8	1.8
Fuyu-8B	0.01	0.14	1.05	0.01	0.14	1.0	0.0	0.04	1.0	0.03	0.0	1.02
InternVL-2.5-8B	8.21	17.02	2.04	5.18	14.64	1.72	6.02	19.3	1.85	5.62	14.03	1.9
Glm-4v-9B	4.65	13.9	2.34	4.25	18.03	2.31	3.09	12.82	2.07	3.83	11.67	2.18
Pixtral-12B	11.88	21.78	2.86	5.88	12.8	2.66	6.5	18.74	2.33	6.32	13.33	2.72
InternVL-2.5-38B	13.71	26.86	3.85	9.51	**24.75**	3.61	11.29	27.46	3.54	8.95	**23.08**	3.62
QwenVL-72B	12.8	26.47	3.37	11.08	22.88	3.22	10.23	30.27	3.05	9.17	18.22	2.99
Claude 3.5 Sonnet	15.14	24.4	**4.62**	**16.25**	19.3	**4.39**	**17.16**	**34.6**	**4.45**	**13.53**	16.21	**4.47**
Gemini-2.0-Flash	**17.02**	**28.54**	3.26	11.34	13.32	3.45	11.98	23.08	3.43	11.09	17.46	3.79
GPT-4o (1120)	13.95	22.94	4.26	10.32	9.93	4.35	12.87	22.63	4.28	11.15	12.81	4.39

Table 7: Results of Web Code Editing on different languages.

Web QA Evaluation Prompt

```
examples = [
   {
      "INPUT": {
         "question": "What is the capital of France?",
         "model_answer": "Paris",
         "ground_truth": "Paris",
      },
      "OUTPUT": {
         "rating": 1,
         "rationale": "The model's answer matches the reference answer exactly."
      }
   },
   {
      "INPUT": {
         "question": "What is in the left of the image?",
         "model_answer": "A bus is in the left of the image.",
         "ground_truth": "A dog is in the left of the image.",
      },
      "OUTPUT": {
         "rating": 0,
         "rationale": "The model's answer is incorrect because the reference answer is 'A dog'."
      }
   },
   {
      "INPUT": {
         "question": "Where is the burger on the table? Tell me the coordinates.",
         "model_answer": "The burger is on the table.",
         "ground_truth": "The burger is on the table at (50, 10, 150, 60).",
      },
      "OUTPUT": {
         "rating": 0,
          "rationale":  "The predicted answer is incomplete because it does not provide the
coordinates as requested in the question."
      }
   }
]
test_case = {
   "INPUT": {
      "question": question,
      "model_answer": model_answer,
      "ground_truth": ground_truth
   }
}
```

You are evaluating a Visual Question Answering (VQA) system's response. Compare the model's answer with the ground truth and rate its accuracy.
Rating Scale (1 or 0):
1 - **Correct and Complete**: - The predicted answer fully matches the ground truth. - No factual errors or missing details. - Addresses the question with the correct level of specificity.
0 - **Incorrect or Irrelevant**: - Any factual errors or mismatches with the reference answer. - Does not address the question properly. - Provides misleading or irrelevant information.
Examples for reference: json.dumps(examples, indent=4)
Question, Model Answer, and Ground Truth: json.dumps(test_case, indent=4)
You must provide your evaluation in the following JSON format (without any extra text): json.dumps("rating": 0 or 1, "rationale": "[Brief explanation of why this rating was chosen]") """

Figure 9: LLM-as-judge prompt for WebQA task using few-shot examples

Mockup2Code Evaluation Prompt

Task Overview: Your task is to evaluate the accuracy of an AI-generated website by comparing it against a provided input sketch. The AI-generated website is provided as an image rendering of the HTML/CSS output. Your goal is to assess how well this rendered image replicates the intended layout from the sketch.
Provided Inputs: You will receive two images:

1. **Input Sketch** – A wireframe illustrating the intended layout.
2. **Predicted AI-Rendered Website Image** – A screenshot of the website generated from AI-created HTML/CSS based on the sketch.

Since the AI-generated website is provided as an image, your evaluation must be based entirely on visual accuracy, disregarding the underlying code implementation.
Step 1: Detailed Description of Both Images
For each image (**Input Sketch** and **AI-Rendered Website**), provide a highly-detailed breakdown based on the following categories. Ensure that descriptions follow the same format for both images to facilitate a precise comparison.
1. Identify All Structural Sections:
Describe in detail the overall structure of the webpage layout, covering the following:

- **Header** – Does it contain a logo, navigation menu, search bar, or other elements?
- **Navigation Bar** – Describe the menu items. How many items are there? Is the navigation horizontal or vertical?
- **Main Content Area** – Identify distinct sections such as hero banners, text areas, images, or interactive components.
- **Sidebars (if applicable)** – Is there a sidebar for additional navigation, filters, or widgets?
- **Footer** – What content is present (e.g., links, social icons, contact information)?

For the AI-rendered website, note any differences compared to the sketch (e.g., missing sections, extra sections, missing items, misplaced content).
2. List and Describe All Elements:
List all key elements present in the **Input Sketch** and **AI-Rendered Website**:

- **Text Elements** – Titles, paragraphs, labels, lists, captions.
- **Images & Icons** – Identify all image placeholders and their intended placement.
- **Buttons & Links** – Describe all interactive elements like CTAs, navigation links, or form buttons.
- **Forms & Inputs** – Search bars, text fields, dropdowns, checkboxes, radio buttons, etc.
- **Tables & Lists** – If present, describe their structure and formatting.

For the AI-rendered website, specify any elements that are missing, added, or incorrectly placed.
3. Layout & Positioning Details:
Describe and analyze the spatial arrangement of elements in both images:

- **Column Structure** – Is the design single-column, multi-column, or grid-based?
- **Alignment** – Are elements aligned left, center, right, or justified?
- **Spacing & Proportions** – Are elements evenly spaced? Are margins, padding, and gaps consistent?
- **Relative Proportions** – Are certain sections (e.g., hero banners, sidebars) larger than others?

For the AI-rendered website, describe any deviations from the sketch (e.g., elements' size differences, elements too large/small, uneven spacing, misalignments).
Step 2: Evaluation of the AI-Rendered Website
After describing both images, evaluate the AI-generated website's accuracy using the following criteria. Assign a score from 1 to 5 for each.
1. Layout Structure Accuracy (1-5):
Does the generated HTML structure strictly follow the wireframe in layout, hierarchy, and element grouping? This includes the correct placement, nesting, and semantic usage of standard structural elements: `<header>`, `<nav>`, `<main>`, `<section>`, `<aside>`, `<article>`, `<footer>`, `<div>`, and content containers like `<img>`, `<p>`.

- **5** → 100% match. All elements are correctly placed, properly nested, fully grouped, and semantically accurate. No missing, misplaced, or extra elements.
- **4** → Mostly accurate, but minor structural inconsistencies exist (e.g., an unnecessary wrapper, slightly misplaced section, or minor redundancy). No missing elements.
- **3** → Some structural errors — at least one missing or misused element, multiple misplaced sections, or noticeable grouping issues.
- **2** → Major deviations — multiple missing, misplaced, or incorrectly nested elements, affecting hierarchy and readability.
- **1** → Severe structural failure — multiple core sections are absent or completely misstructured, making the output unrecognizable compared to the wireframe.

2. Spacing & Proportions (1-5):
Do margins, paddings, and element dimensions (e.g., `width`, `height`, `max-width`, `min-width`, `max-height`, `min-height`, gap for flex/grid layouts) precisely match the wireframe?

- **5** → 100% correct. All elements have precise margins, paddings, widths, heights, and spacing. No deviations.
- **4** → Minor inconsistencies exist (e.g., slightly incorrect padding/margin values or minor width/height variations).
- **3** → Noticeable discrepancies — some elements are too large, too small, or unevenly spaced, affecting visual balance.
- **2** → Significant spacing issues — multiple elements have incorrect dimensions, margins, or paddings, leading to a visibly distorted layout.
- **1** → Severe inaccuracies — most elements have incorrect proportions or spacing, making the layout visually broken and inconsistent with the wireframe.

3. Alignment & Grid Consistency (1-5):
Are elements precisely aligned according to the wireframe, following the expected grid/flex structure and ensuring uniform positioning?

- **5** → Perfect alignment. Every element follows the wireframe's grid, flex, or positioning structure exactly. No misalignments.
- **4** → Mostly aligned, but minor deviations exist (e.g., slightly off-center text or small pixel variations in placement).
- **3** → Some clear misalignments — at least one noticeably off-grid or misplaced element that affects overall balance.
- **2** → Major alignment issues, with multiple elements misaligned, overlapping, or not following the expected structure.
- **1** → Severe disorganization — the output fails to follow the wireframe's grid or positioning, making the layout appear chaotic.

Final Score Calculation:
Final Score = (Layout Structure Accuracy + Spacing & Proportions + Alignment & Grid Consistency) / 3
Output Format:
Your response must follow this JSON structure:

```
{
  "descriptions": {
    "input sketch": "provide the description of sketch here",
    "AI-rendered website": "provide the description of website here"
  },
  "scores": {
    "layout_structure_accuracy": [1-5],
    "spacing_proportions": [1-5],
    "alignment_grid_consistency": [1-5]
  },
  "final_score": [calculated average score],
  "reasoning": "[Concise evaluation highlighting key strengths and weaknesses]"
}
```

Figure 10: LLM-as-Judge input prompt: It evaluates the model output and the ground truth among some detailed criteria given in the prompt.

Web Code Editing Evaluation Prompt

You are evaluating a system that generates HTML code based on a given task. Compare the predicted code with the ground truth code and rate its correctness based on functionality rather than exact syntax. If the code performs the intended task correctly, even if formatted differently or using a different approach, it should receive a high score.
Rating Scale:

- 5 - PERFECT - Fully achieves the required functionality as described in the reference output. - May have differences in syntax or structure, but effectively performs the same task with no missing elements.
- 4 - CORRECT BUT WITH MINOR ISSUES - Achieves the intended functionality but has small flaws (e.g., slightly different behavior, minor inefficiencies).
- 3 - PARTIALLY CORRECT - Achieves part of the intended functionality but is missing key aspects or has notable issues.
- 2 - MOSTLY INCORRECT - Fails to accomplish most of the required functionality but shows some partial effort.
- 1 - COMPLETELY INCORRECT - The solution does not fulfill the required functionality or is entirely off-target.

Examples for reference:

```
examples = [
    {
      "INPUT": {
        "question": "Change the header's background color to blue.",
        "model_answer":"+<style>header{background-color:blue;}</style> <header>Welcome</header>",
        "ground_truth": "<header style='background-color: blue;'>Welcome</header>"},
      "OUTPUT": {
        "rating": 5,
        "rationale": "The model answer correctly implements the change by ensuring the header
displays with a blue background. Despite using a style tag in the model answer versus inline
styling in the ground truth, both approaches deliver the exact intended functionality." }
    } ]
```

Task for Evaluation:

```
    {
      "INPUT": {
        "question": "<question>",
        "model_answer": "<model_answer>",
        "ground_truth": "<ground_truth>"
      }
    }
```

Provide your evaluation in the following JSON format (using json delimiters, do not include any extra text):

```
    {
      "rating": "1 or 2 or 3 or 4 or 5",
      "rationale": "[Brief explanation of why this rating was chosen]"
    }
```

Figure 11: LLM-as-judge prompt for Web Code Editing task using few shot examples.

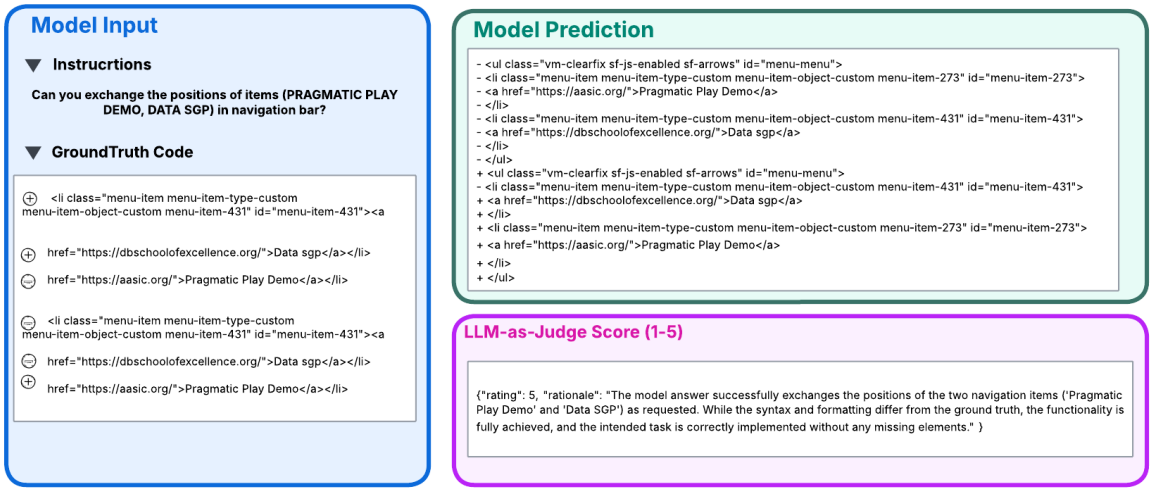


Figure 12: Code edit case study: **Success (Claude-3.5)**. The generated code accurately addresses the instructions given as input.

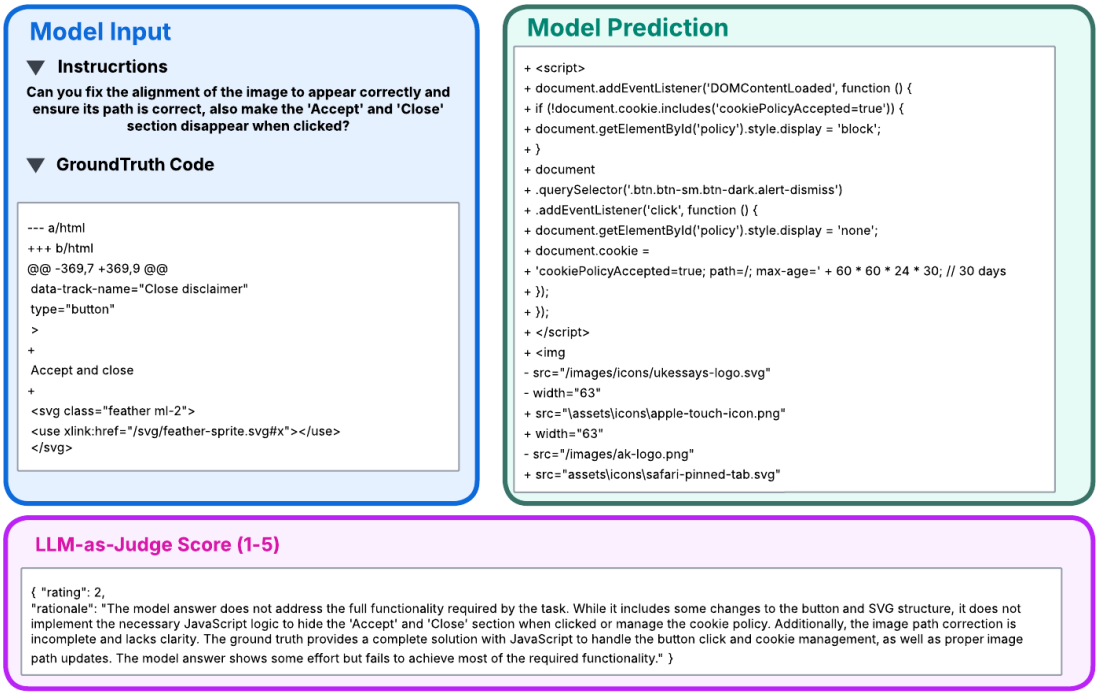
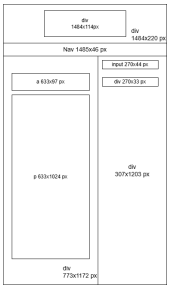


Figure 13: Code edit case study: **Failure (Gemini-2.0-Flash)**. The generated answer by the model skips main modifications requested by the user.

OpenAI-o1

Original Page



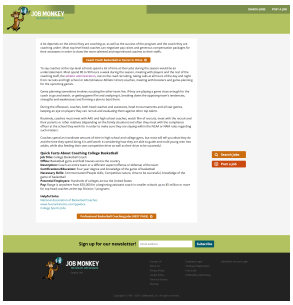
Mockup Image

Generation

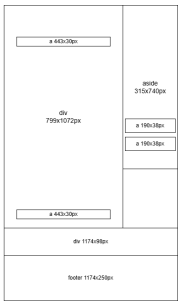
GPT Score Evaluation

```
Alignment: 2. Key elements (e.g., input box) misaligned, deviating from intended grid.
Layout: 2. Two-column structure poorly represented, essential sections missing/merged.
Spacing: 2. Uneven element distribution results in inconsistent spacing and imbalance.
Overall Score: 2
```

InternVL-38B



Original Page



Mockup Image

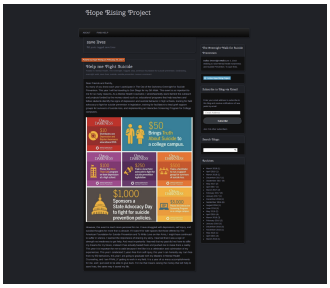
Generation

GPT Score Evaluation

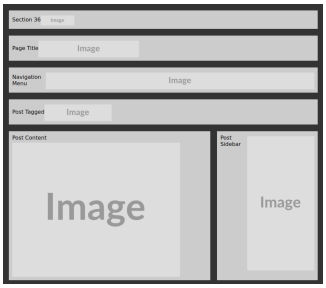
```
Alignment:
    2. The main content is misaligned; also, key elements like the aside and buttons are missing.
Layout: 2. The intended two-column structure is replaced by a single-column layout, significantly
    deviating from the sketch.
Spacing: 2. Inconsistent spacing and proportions arise from the absence of the aside and smaller
    placeholders.
Overall Score: 2
```

Figure 14: Examples of the **failure cases on the Mockup2Code task** for the best closed-source model (OpenAI-o1) and the best open-source model (InternVL2.5-38B).

OpenAI-o1



Original Page



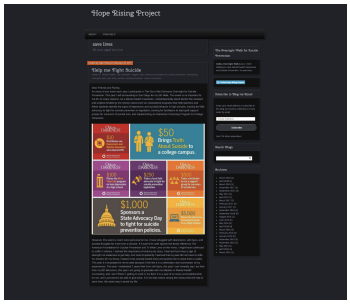
Mockup Image

Generation

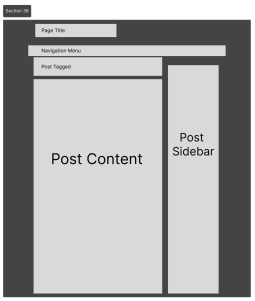
GPT Score Evaluation on Simple sketch

Layout: 5. Structure perfectly replicated.
Spacing: 4. Minor inconsistencies in spacing.
Alignment: 5. Precise alignment maintained.
Overall Score: 4.67

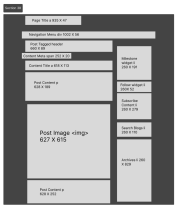
OpenAI-o1



Original Page



Mockup Image



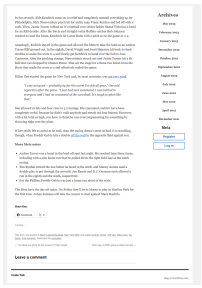
Generation

GPT Score Evaluation on Complex Sketch

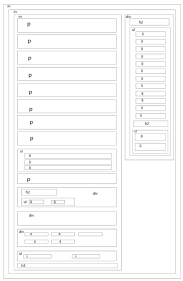
Layout: 5. Structure and sections perfectly replicated.
Spacing: 5. Spacing and proportions are consistent.
Alignment: 5. Alignment is precise and matches the sketch.
Overall Score: 5

Figure 15: Examples of the **success cases on the Mockup2Code task** for the best closed-source model (OpenAI-o1) for both simple and complex mockups.

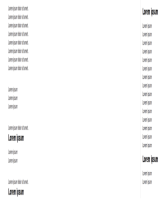
Internvl2.5-8b



Original Page



Mockup Image

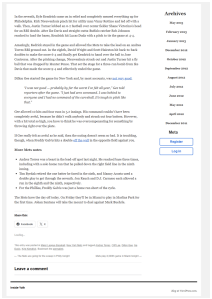


Generation

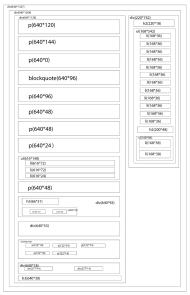
GPT Score Evaluation on Simple Sketch

```
Layout: 2. Many key sections, headers, and lists are missing, leading to a poor structural match with
      the sketch.
Spacing: 2. Uneven gaps and inconsistent spacing cause improper grouping and distorted proportions.
Alignment: 2. Misaligned elements and an irregular grid result in a disorganized layout.
Overall Score: 2
```

Internvl2.5-8b



Original Page



Mockup Image

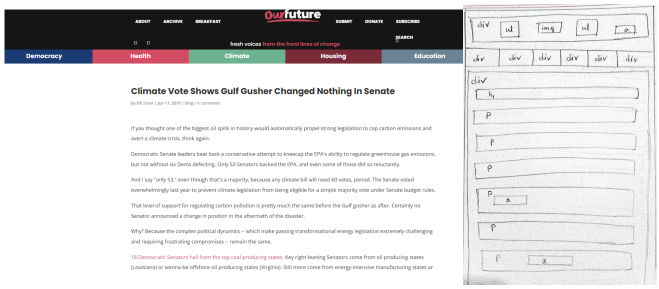
Generation

GPT Score Evaluation on Complex Sketch

```
Layout: 4. Two-column layout is largely reproduced with most elements accurately placed, though an
     extra image placeholder appears.
Spacing: 3. Inconsistent spacing leads to uneven gaps that affect the overall balance.
Alignment: 3. Some elements are misaligned, disrupting the intended grid structure.
Overall Score: 3.33
```

Figure 16: Examples of the **failure cases on the Mockup2Code task** for the open-source model (Internvl2.5-8b) for both simple and complex mockups.

OpenAI-o1

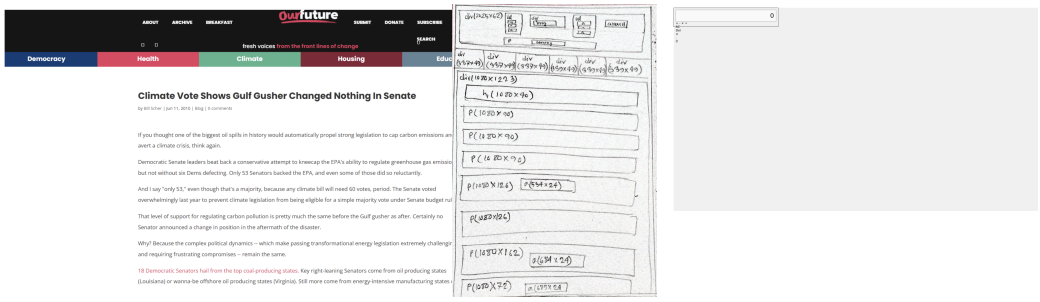


Original Page **Mockup Image** **Generation**

GPT Score Evaluation

Alignment: 2. Key elements (e.g., input box) misaligned, deviating from intended grid.
Layout: 2. Two-column structure poorly represented, essential sections missing/merged.
Spacing: 2. Uneven element distribution results in inconsistent spacing and imbalance.
Overall Score: 2

OpenAI-o1



Original Page **Mockup Image** **Generation**

GPT Score Evaluation

Alignment:
2. The main content is misaligned; also, key elements like the aside and buttons are missing.
Layout: 2. The intended two-column structure is replaced by a single-column layout, significantly deviating from the sketch.
Spacing: 2. Inconsistent spacing and proportions arise from the absence of the aside and smaller placeholders.
Overall Score: 2

Figure 17: Examples of the **failure cases on the Mockup2Code task** for the best closed-source model (OpenAI-o1) for both simple and complex mockups.