

HOW FRAGILE ARE GUI AGENTS? A CONTROLLED PERTURBATION BENCHMARK FOR INTERFACE EVOLUTION

ZUCHEN LI
zuchenli@umich.edu

OPEYEMI AKINNIYI
akinniyi@umich.edu

XIANGNONG WU
xiangnwu@umich.edu

CAROL KANG
yhankang@umich.edu

HAO YIN
haoyin@umich.edu

I. MOTIVATION & RESEARCH QUESTIONS

Vision-Language Model (VLM) based GUI agents promise to automate workflows across the live web, but real-world interfaces are **not static**: buttons rename, layouts reflow, forms restructure, and entire workflows change between releases. An agent that works today may silently fail tomorrow. *Temporal robustness* of GUI agents remains largely uncharacterized.

Existing benchmarks (VisualWebArena, WebArena, Mind2Web) evaluate agents on **fixed version** of websites. They tell us how well an agent performs on *one version* of an interface, but not whether the agent’s competence is robust to the kinds of UI changes that happen in the wild.

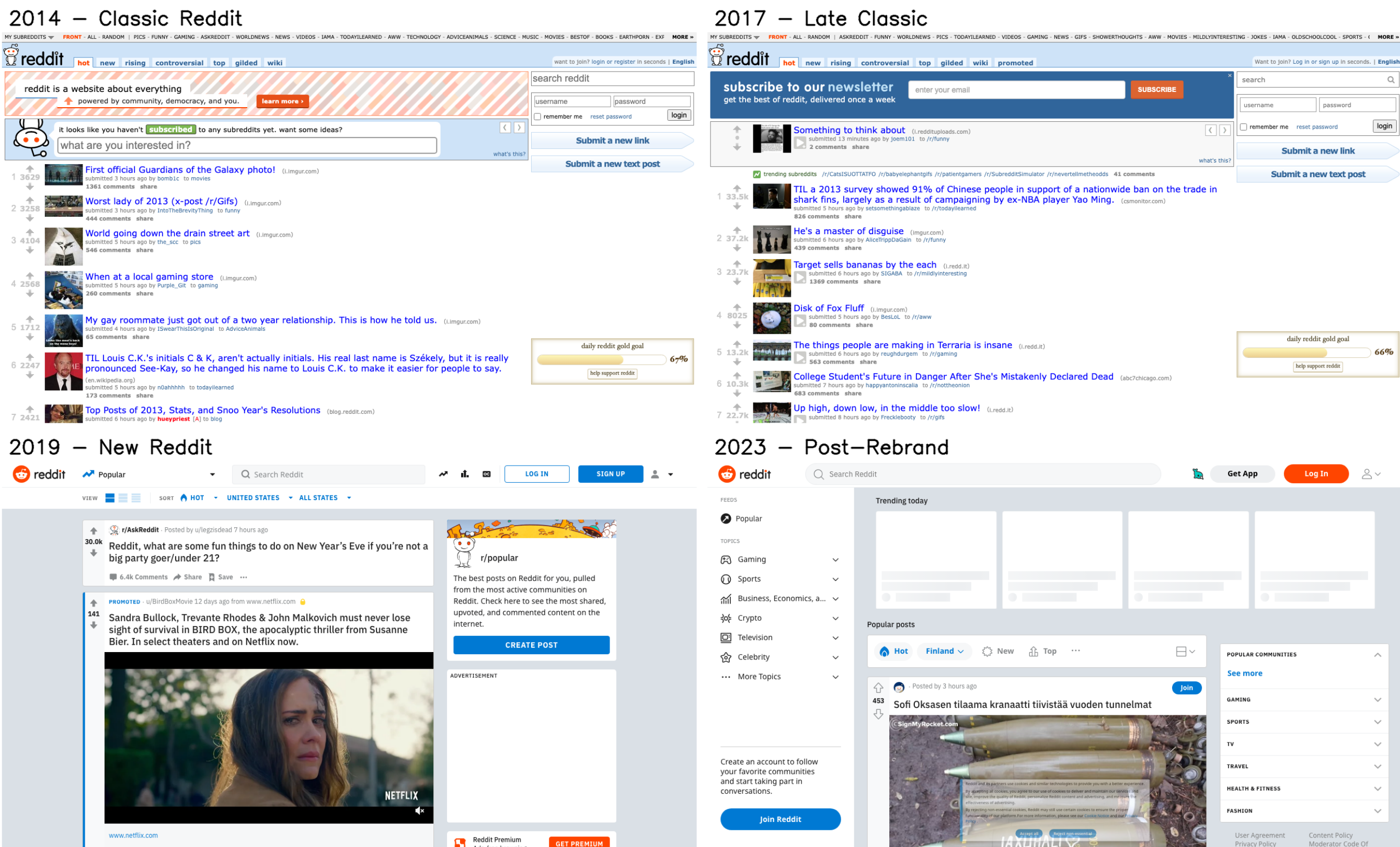
Core Research Questions

- How much do VLM-based GUI agents degrade under *controlled* interface perturbations, and which perturbation types hurt the most?
- Can lightweight *test-time interventions* (experience memory, reasoning guidance) recover lost performance without retraining?
- What *failure modes* characterize non-robust GUI agents, and what design principles do they suggest?

Contributions. (i) A controlled, synthesis-based benchmark for GUI agent robustness with a principled perturbation taxonomy; (ii) comparative evaluation of three agent strategies under matched perturbations; (iii) failure-mode analysis yielding actionable design insights.

II. PRELIMINARY STUDY: WAYBACK MACHINE

We first asked whether **real-world archived snapshots** could serve as a temporal robustness testbed. We collected Wayback Machine captures of *Reddit*, *IMDb*, and *Craigslist* spanning **2014–2023**



Preliminary finding and takeaway Agent performance **fluctuates** across snapshot years, but the pattern is inconsistent. To make temporal robustness measurable, we need *controlled* perturbations with known type, location, and magnitude. This motivates Column II.

III. CONTROLLED PERTURBATION BENCHMARK

We move from **archived** to **synthesized** websites: using LLM-driven web synthesis (“vibe coding”), we generate *base sites* in multiple visual styles and families of *perturbed variants* derived from each base, every variant differing from its base by a **single, typed, labeled** change. Agents are evaluated within the *VisualWebArena (VWA)* framework, which provides a standardized multi-step agent loop with SoM grounding and automated success evaluation.

Testbed: Job Application Site. Multi-step interactive tasks — browse roles, filter listings, fill application, submit — over a lightweight Flask backend, exercising *workflow* robustness: state must be tracked across screens and form actions composed without error.

Perturbation taxonomy. Every perturbed variant is tagged with one of five types at one of three intensity levels (L1 subtle → L3 aggressive):

Type	Example change
LAYOUT	single-column → two-column → card grid
SEMANTIC	“Contact” → “Get in Touch” → “Reach Out”
VISUAL	text link → iconified button → floating CTA
STRUCTURAL	inline content → collapsed accordion → modal
WORKFLOW	single form → 2-step wizard → 4-step wizard

Generation methodology. The testbed is realized as a family of *base sites* produced by prompting a coding-capable LLM from a fixed content specification (role listings, application materials, and form fields). To prevent overfitting to a single aesthetic, bases are generated in **multiple visual styles**. *Perturbed variants* are then derived from each base by instructing the LLM to apply a **single, typed** modification while holding all content fixed, so within a style only the axis of interest varies.



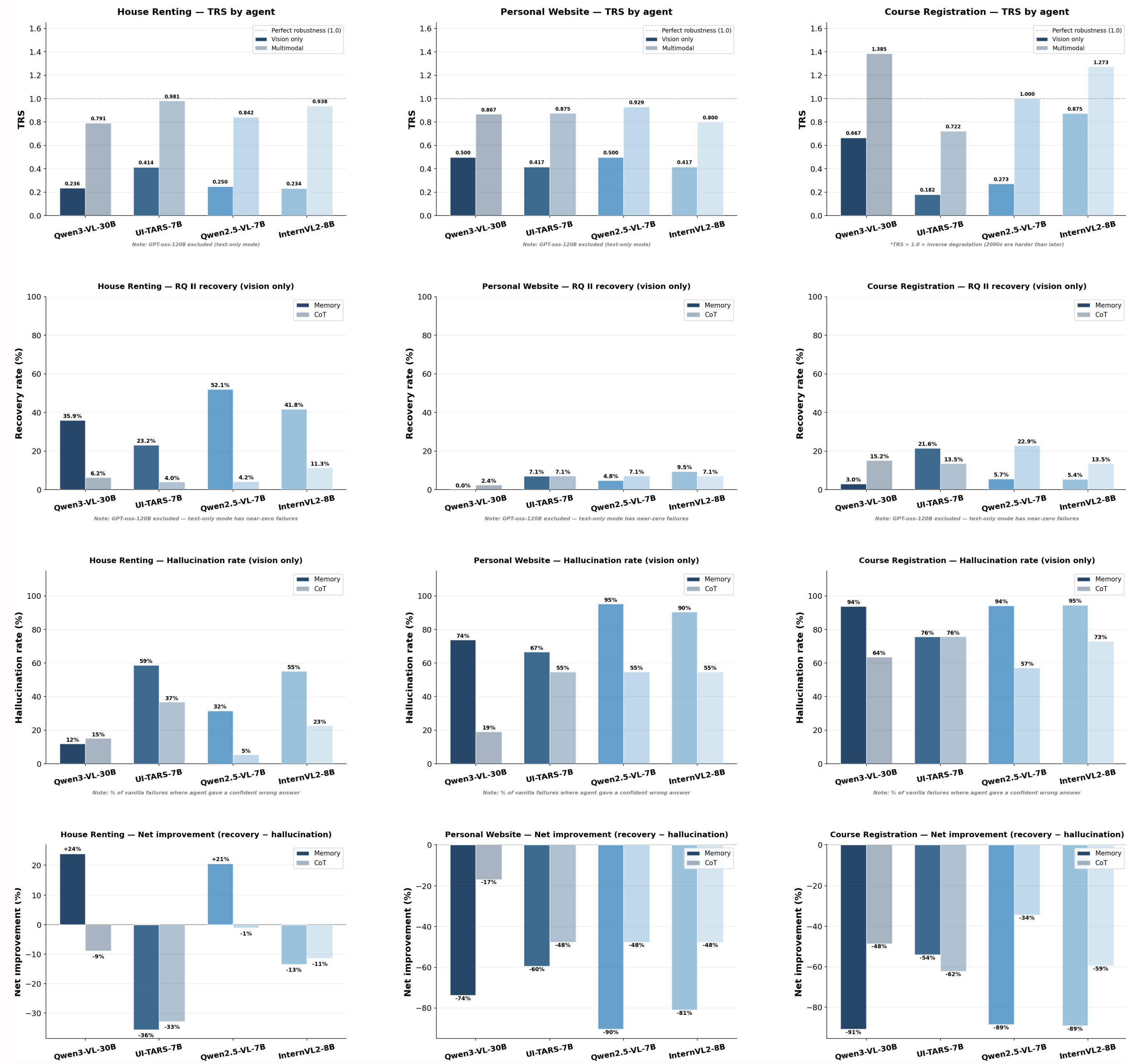
Fig. 1: Three visual-style realizations of the Job Application testbed.

Task & evaluation design. Each task in `tasks.json` specifies an intent, a start URL, and an automated evaluator (URL match, DOM-element match, or LLM judge). Tasks span three difficulty tiers: *extraction* (“find the application deadline for the open role”), *navigation* (“locate the hiring FAQ and report the contact email”), and *multi-step submission* (“apply to the open role using the provided CV and cover letter”). Every task is validated to **succeed on the base** before deployment on perturbed variants, ensuring any failure attributes to the perturbation, not the task.

- Synthesized sites are *unseen* by pretrained VLMs ⇒ clean generalization signal.
- Single-axis perturbations *isolate* the causal factor behind each failure.
- Typed + leveled labels make difficulty *quantitative*, not subjective.

IV. EVALUATION RESULTS

We first evaluated 4 agents (Qwen3-VL-30B, UI-TARS-7B, Qwen2.5-VL-7B, and InternVL2-8B) with WebVisualArena. All three agents evaluated share the same underlying VLM and observation format; they differ only in **test-time prompting strategy**, isolating the effect of the intervention itself.



Rows 4 5 break down hallucination: for example, memory + CoT yields 36% recovery but 12% hallucination, resulting in a 24% net improvement.

V. PRELIMINARY FINDINGS

Key Findings

- Agents are highly **UI-fragile**: the *same data* in a modern click-to-expand layout drops success from 85.0% to 42.9% ($\Delta = 42\%$); 90s-style flat HTML (88.2%) beats a Notion-style UI (70.6%) by 17 points.
- Failures cluster into four modes: *hidden-content blindness* (no click to expand), *format mismatch* (right value, wrong string), *multi-step interaction* (filter → select → read), and *stall-and-give-up* (repeated actions).

Proposed test-time fixes

- Explore-before-stop*: block the stop action until every tab/toggle has been tried.
- Format-quote verification*: force the agent to quote the exact page substring before answering.
- Backtrack-on-stall*: detect repeated actions and inject an alternative-path hint.