Cua-Bench: Technical Report

Cua AI Team

https://cuabench.ai/

Abstract

Deploying autonomous computer-use agents (CUAs) requires robust training and evaluation across diverse and realistic computing environments. Existing benchmarks largely rely on static virtual machines with fixed applications and narrowly defined tasks, failing to capture the variability in operating systems, application states, GUI layouts, and desktop contexts encountered in real-world deployment. We introduce **Cua-Bench**: a flexible and scalable framework for constructing verifiable, dynamic computer-use environments that support comprehensive CUA training and evaluation. Cua-Bench provides CLI workflows for GUI/UI data generation, agentic trajectory generation, verifiable task generation and evaluations. Cua-Bench not only provides simulations of real applications and computing environments, but it also enables diverse data creation and agentic testing across heterogeneous GUI and system configurations. The framework exposes a Python API for defining task setup, evaluation, and oracle solutions and evaluation atop web-framework-agnostic desktop GUI applications, making it easy for researchers and LLMs to develop and extend new tasks across UI/GUI data generation, trajectory synthesis, and simulation use cases. This modularity allows developers to easily integrate preferred tooling to build data generators, tasks, and simulators, making Cua-Bench a practical foundation for realistic, large-scale CUA benchmarking and development.

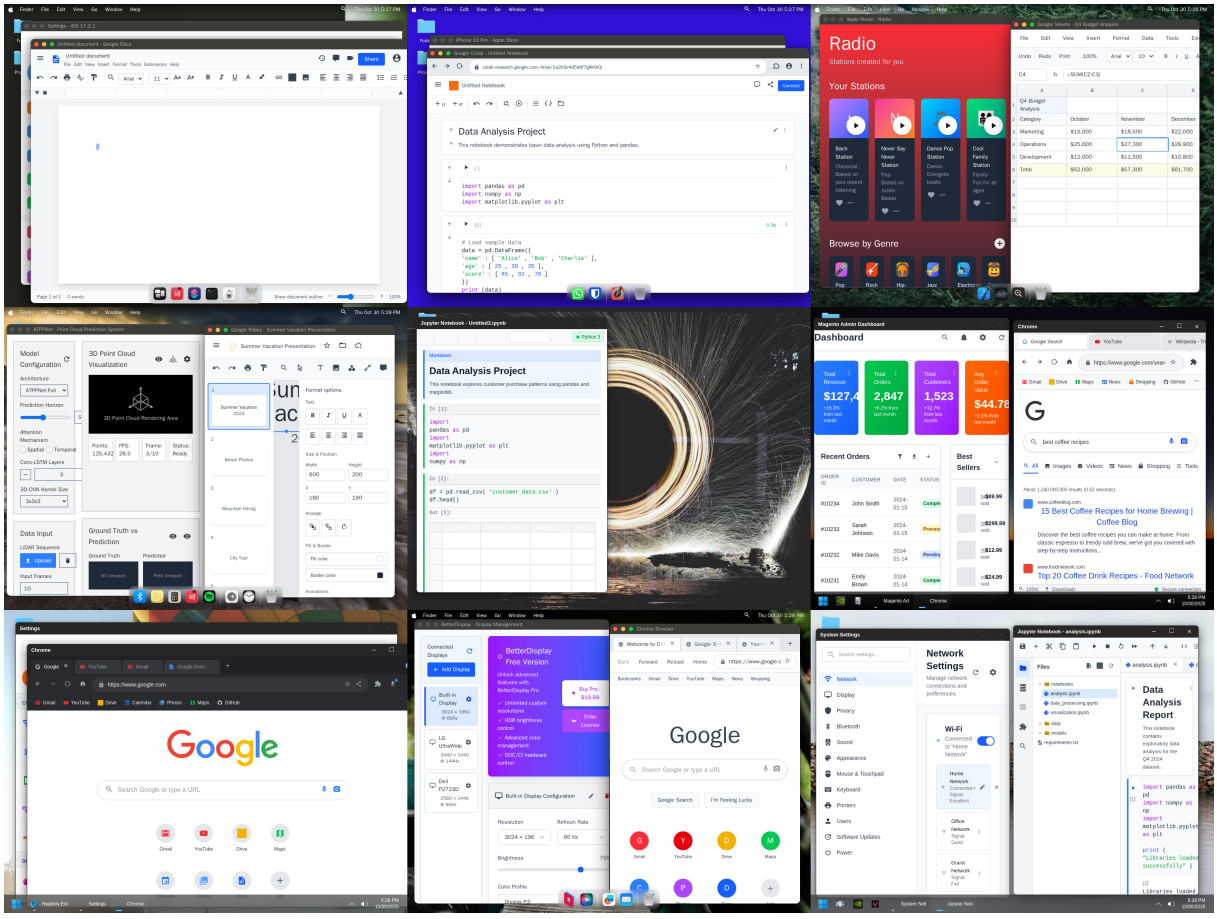

Figure 1: Examples of data generated by Cua-Bench

1 Introduction

Recent advances in foundation models have given rise to a new class of autonomous agents capable of directly interacting with desktop computing environments: clicking buttons, typing text, navigating applications, and completing complex multi-step workflows.

However, across the diverse computing environments agents encounter in testing and deployment, they face significant variation in across multiple dimensions. To name just a few:

- Operating systems: Windows 10 or 11, macOS versions from Mountain Lion to Sequoia, various Linux distributions with different desktop environments and distros (GNOME, XFCE, KDE).
- Desktop states: Different sets of installed applications, varying favorited and pinned apps, diverse wallpapers and themes.
- Window layouts: Multiple open windows at various positions, overlapping applications, cluttered desktops.
- Application configurations: Light/dark themes, different screen resolutions, varying font sizes.

Current evaluation frameworks, however, largely rely on static virtual machine snapshots with fixed configurations. Benchmarks like OSWorld and Windows Agent Arena provide valuable evaluation infrastructure but suffer from critical limitations such as:

- Static VMs: Tasks are baked into bench-server components embedded in VM images, requiring up to 20 minutes to load.
- Fixed application sets: VMs come preinstalled with a predetermined set of applications, not reflecting real-world diversity.
- Limited task definitions: Tasks are defined via JSON with constrained setup and evaluator command vocabularies specific to each benchmark.
- Slow iteration: Updating tasks requires rebuilding entire VM images: a slow process.

1.1 Capabilities & Features

Cua-Bench provides a scalable and customizable framework to help with agentic data, testing, evaluation, and more. To list just a few features, it allows users to:

1. Generate thousands of configurable desktop variations across macOS, Windows, and Linux, including randomized installed applications, favorited apps, and window layouts.
2. Define tasks with verifiable rewards through a declarative Python API supporting custom JSON scenarios for configurable application variations.
3. Deploy lightweight "webtop" environments—HTML/CSS-based desktop replicas—requiring only a single CPU without virtualization infrastructure.
4. Generate ground-truth multi-step trajectory data through reference solutions, inspired by Terminal-Bench's oracle-based approach. We also introduce a novel HTML-based snapshot format that enables offline rendering, domain-specific augmentation, and cross-OS compatibility.

Cua-Bench supersedes prior frameworks by decoupling tasks and data generation from VM images (though we also include this option if necessary), enabling rapid iteration without rebuilding slow disk images, and providing an agnostic Python API for environment and window control.

	macOS	**Linux**	**Windows**	**Android**	**iOS**	**VM**	**Webtop**
UI & GUI Data Generation	✓	✓	✓	✓	✓	✓	✓
Trajectory Data Generation	✓	✓	✓	✓	✓	✓	✓
Agentic Benchmarks	✓	✓	✓	✓	✓	✓	✓
Shell Apps & Simulators	✓	✓	✓	✓	✓	✓	✓
Real Apps	✓	✓	✓	✓	✓	✓	×

Table 1: Overview of features & capabilities supported within Cua-bench across different environments and platforms. Cua-Bench runs macOS, Linux, Windows, Android, and iOS in VM or web-based desktop environments, offering a Python API to deploy HTML-based GUI apps across both, while native binary applications are supported in VM environments.

2 Scalable Data Generation & Diversity

2.1 GUI Data & HTML Snapshots

Cua-Bench is capable of generating realistic and diverse GUI data at scale—customizable and configurable to generate data across multiple dimensions such as different programs and applications, window placements and screen coverage, different graphic styles, colors, and contrasts, different platforms/devices, and much more (Figure 1). Beyond raw screenshots, Cua-Bench captures full HTML snapshots of each window along with bounding box coordinates, accessibility labels, and CSS styles, enabling offline rendering and cross-OS replay of captured states (Figure 2).

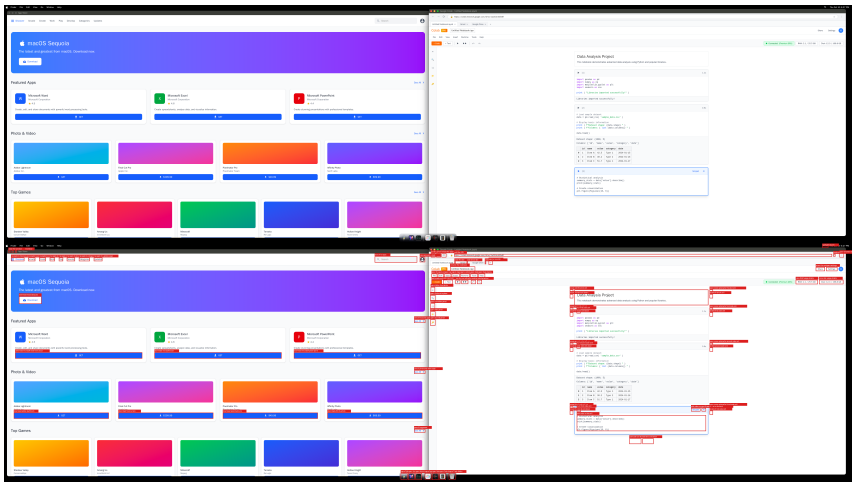

Figure 2: Examples of Cua-Bench GUI data, containing high-quality UI/GUI screenshots (top row) and meta-data such as bounding boxes and accessibility labels around UI elements (e.g., bottom row), etc.

2.1.1 Variety Across Platforms

In terms of cross-OS and cross-platform variety, Cua-Bench is able to generate data at scale across multiple platforms and device types, with each environment optionally seeded with hundreds of real-world applications and icons to simulate natural desktop clutter. Beyond conventional desktop computing environments (e.g., Windows, Mac, Linux), Cua-Bench is also capable of generating data of GUI/UI interfaces from popular mobile environments such as iOS and Android (e.g., Figure 4, Figure 5).

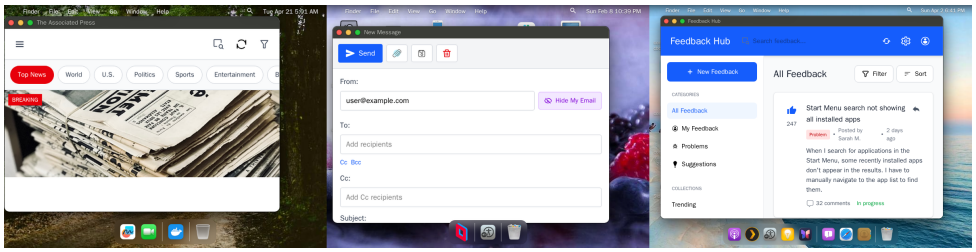

Figure 3: Examples of different MacOS interfaces generated via Cua-Bench with simulated clutter

2.1.2 Variety Across Time

While many GUI, UI, screenshot, and other similar datasets for grounding CUAs in visual computing environments exist with varying degrees of coverage across different computing and operating environments, nearly all of them focus on modern interfaces and modern graphic styles. This, however, can introduce several problems with data representation, potentially biasing and over-fitting agents towards certain GUI/visual environments, resulting in a lack of robustness across diverse settings.

To ensure that agents are universally capable of understanding and handling any type of different graphic/visual environments and styles, Cua-Bench is capable of generating a data from a variety of different OS versions from old to new (e.g., Figure 6) from older to newer versions of Windows OS and others as well as different distributions (e.g., Linux) [1].

[1] Additional OS, environments, and platforms are continuously being incorporated into Cua-Bench to ensure

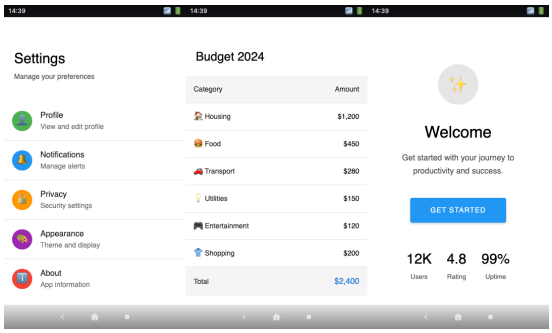

Figure 4: Examples of generated Android data

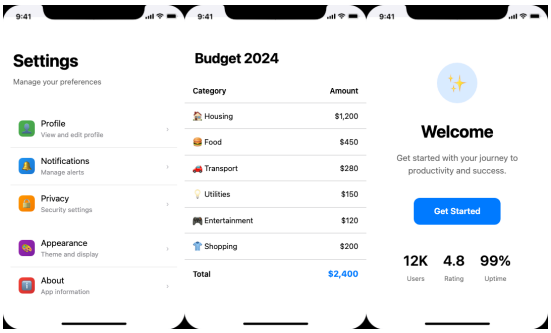

Figure 5: Examples of generated iOS data

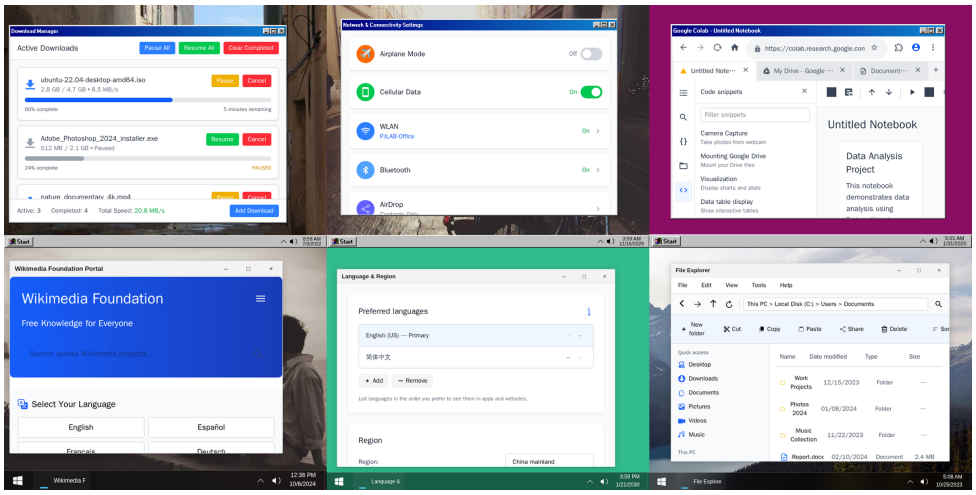

Figure 6: Examples of generating GUI/UI styles across new and old OS environments alike with Cua-Bench (Top: Windows 98, Bottom: Windows 10).

2.1.3 Variety Across Resolutions

Similarly, Cua-Bench is capable of generating GUI data of differing resolutions, allowing for a wide range of detail levels for agents to use for training and testing on rich GUI elements in both high and low resolutions alike (Figure 8, Figure 7).

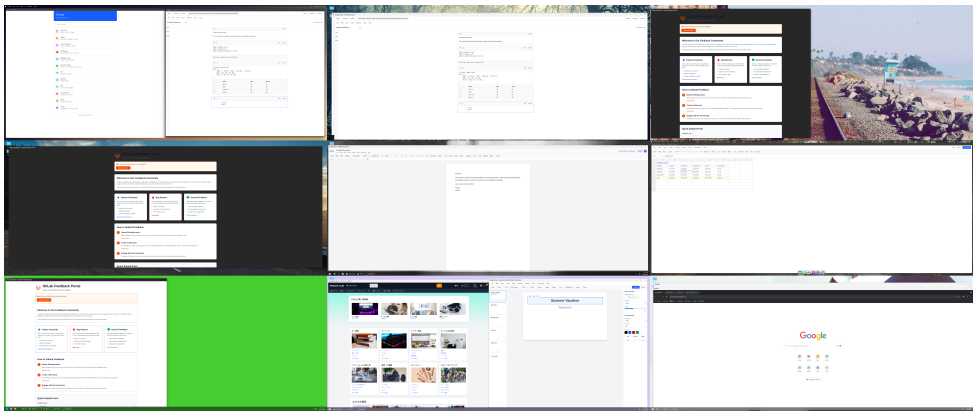

Figure 7: Examples of higher resolution (3440x1440) images generated by Cua-Bench

2.2 Agentic Trajectory Data & Task Generation

Taking advantage of the HTML-based GUI snapshots, Cua-Bench exposes a Playwright-like Python API that tasks can use to define oracle solutions—inspired by Terminal-Bench's solution script approach. This

diverse representation across time and platforms/environments.

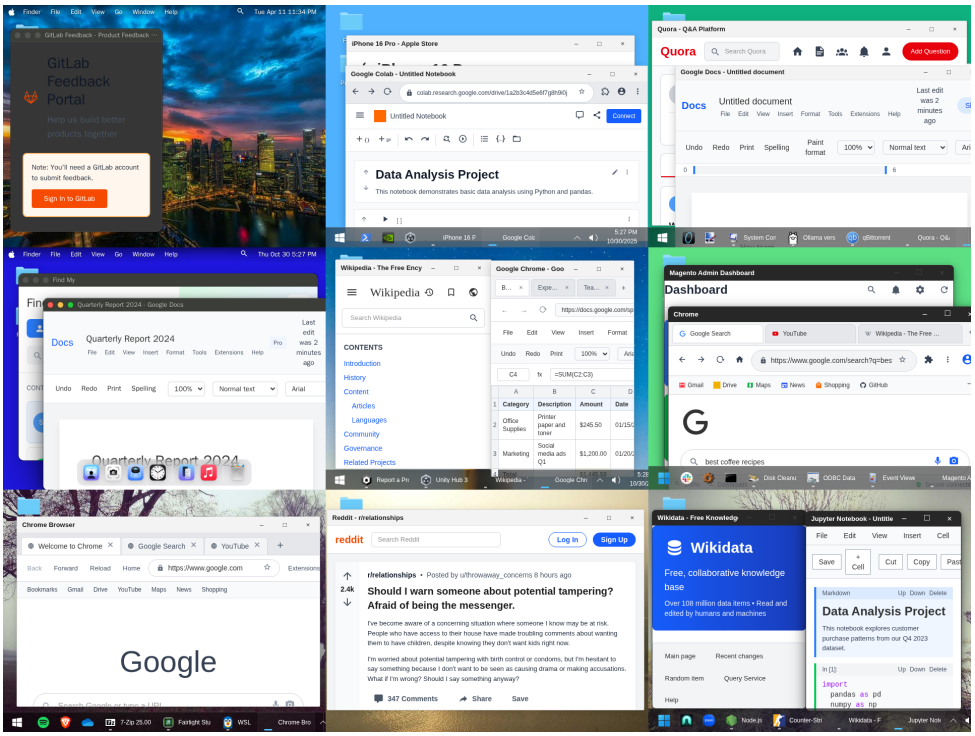

Figure 8: Examples of lower resolution (640x480) images generated by Cua-Bench

allows Cua-Bench to generate agentic trajectories (i.e., "traces") by executing the oracle solution and mapping the high-level Playwright-like code to step-by-step snapshots and low-level keyboard/mouse actions, making it easy to generate many-step ground-truth trajectories for training computer-use agents to navigate complex desktop environments.

2.2.1 Oracle Solutions for Trajectory Generation

Each task in Cua-Bench can define a reference solution using a `@cb.solve_task` decorator, which programmatically completes the task using Playwright-style selectors and actionability logic. When executed, Cua-Bench records each action alongside the environment state—capturing HTML snapshots, screenshots, and input events—to produce complete multi-step trajectories suitable for behavioral cloning or supervised learning (Figure 9, Figure 10).

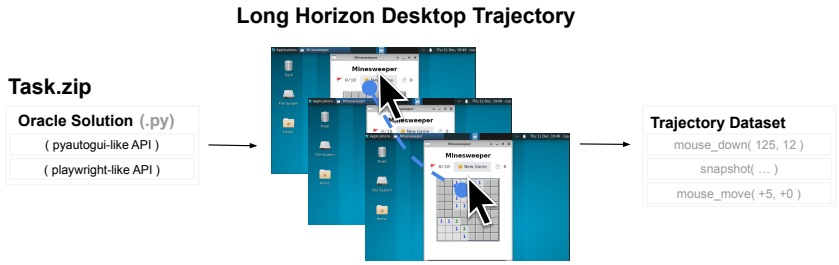

Figure 9: Example of generating a multi-step long-horizon task and collecting its trajectory data in Cua-Bench

2.2.2 Task Generation & Development

Cua-Bench's simple-to-use Python API (both web framework agnostic and supporting Playwright syntax on desktop environments) makes generating verifiable and solvable tasks at scale easy for both humans and modern LLMs acting as task authors. Users can define tasks via four decorators (`@tasks_config`, `@setup_task`, `@evaluate_task`, `@solve_task`), with JSON-based scenario injection enabling thousands of task variations and unique trajectories from a single template (Figure 11, Figure 12).

Figure 10: Examples of trajectory traces from Cua-Bench within a Linux VM environment, containing low-level actions and full HTML snapshots of each window

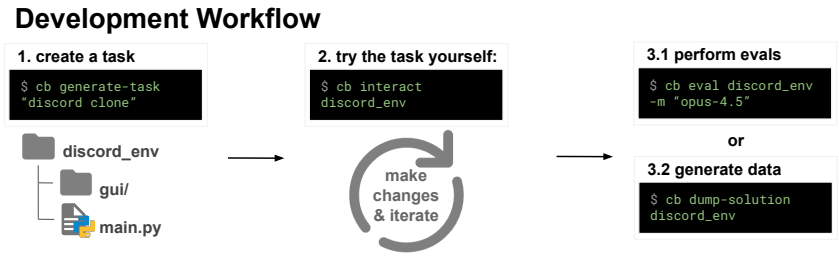

Figure 11: Illustration of development workflow within Cua-Bench to generate tasks, perform evals, and collect trajectory data

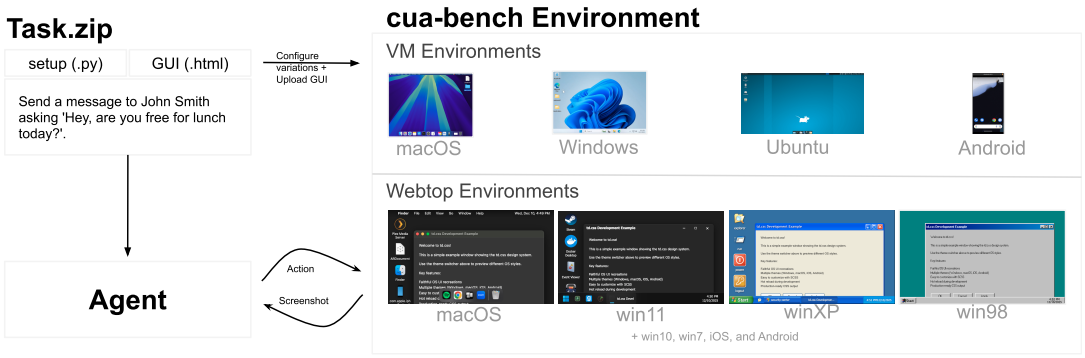

Figure 12: Overview of Cua-Bench's task and environment architecture

3 Simulators & Environments

Beyond just data generation, Cua-Bench offers self-contained, full-bodied simulators and environments for agents to interact with and operate within.

3.1 CUA Benchmark Adapters

The flexibility of Cua-Bench's Python-based task definition API allows existing computer-use benchmarks to also be wrapped as adapters, delegating to their original setup and evaluation code while gaining access to Cua-Bench's variation and recording capabilities. Currently included adapters support OSWorld (Xie et al., 2024), Windows Agent Arena (Bonatti et al., 2025), and MiniWoB++(Liu et al., 2018) though future development efforts will include others as well.

3.2 Shell Applications & Websites

In addition to widely-used CUA benchmarks like Windows Agent Arena and OSWorld, Cua-Bench also provides simulated shell applications with realistic GUI elements in full-fledged environments that agents

can interact with (e.g., Figure 13), allowing agents to easily explore in realistic settings. Cua-Bench also enables data collection from agents' interactions with these applications, allowing the user to generate agent trajectories for later use.

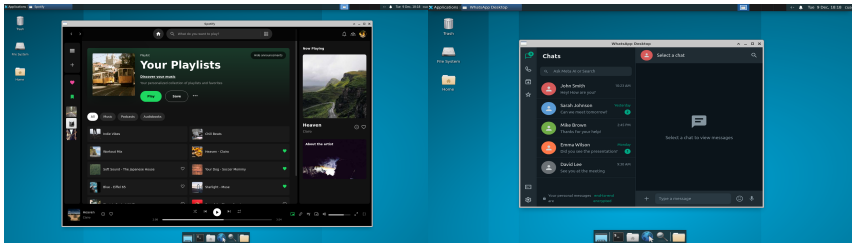

Figure 13: Shell applications within Cua-Bench have full functionality, serving as a robust simulator/environment for an agent interaction and testing.

Cua-Bench also allows for extensive customization of these simulated shell applications, allowing users to customize not just the appearance (e.g., style, color, etc.) and placement of these applications but also the content of these applications as well (Figure 14). As demonstrated by Ullrich et al. (2025), such variation is essential for measuring and improving agent reliability, as task success rates can differ by over 10× depending on theme, font, or language settings.

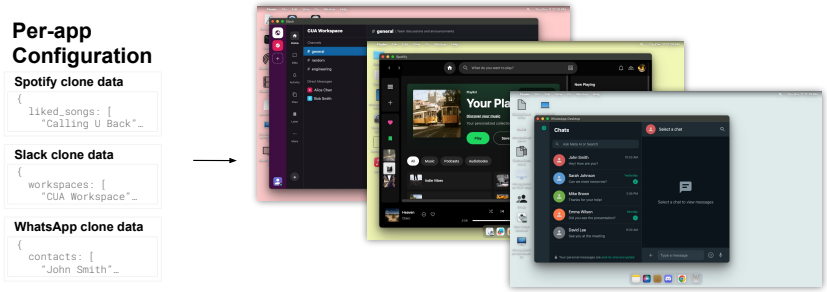

Figure 14: Each application itself is flexible and configurable, allowing for customizing of the types of content, UI elements, and more within the app.

4 Conclusion

The promise of autonomous computer-use agents hinges on their ability to operate reliably across the messy, heterogeneous environments of real-world computing. We have introduced Cua-Bench, an open framework designed to help researchers easily develop controlled benchmarks that are close to deployment reality. Cua-Bench enables researchers and practitioners to define verifiable tasks through a simple Python API, generate thousands of unique desktop configurations, and produce ground-truth trajectories via oracle solutions that simultaneously verify task solvability, removing the development overhead of building virtual machine images and managing virtualization infrastructure. By capturing full HTML snapshots alongside screenshots and input events, Cua-Bench supports offline analysis, augmentation, and cross-platform compatibility that existing OS VM benchmarks cannot provide.

Limitations and future work. Cua-Bench's Webtop environments, while visually realistic and fully interactive, represent simplified versions of complex software and may not capture all edge cases present in real applications. Current task coverage focuses on common desktop workflows; expanding to specialized domains (e.g., creative software, enterprise tools) remains ongoing work. Future directions include integrating reinforcement learning training loops directly into the framework, supporting large-scale human-in-the-loop annotation pipelines, and establishing community-driven task repositories to accelerate progress toward truly general-purpose computer-use agents.

References

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows agent arena: Evaluating multi-modal OS agents at scale. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4874–4910. PMLR, July 2025. URL https://proceedings.mlr.press/v267/bonatti25a.html.

EZ Liu et al. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR) Workshops*, 2018. Includes the MiniWoB++ benchmark suite.

Karen Ullrich, Jingtong Su, Claudia Shi, Arjun Subramonian, Amir Bar, Ivan Evtimov, Nikolaos Tsilivis, Randall Balestriero, Julia Kempe, and Mark Ibrahim. Openapps: Simulating environment variations to measure ui-agent reliability, 2025. URL https://arxiv.org/abs/2511.20766.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, 2024. doi: 10.52202/079017-1650. NeurIPS 2024.