UI-TARS Desktop Agent

This report evaluates the UI-TARS Desktop agent, a GUI-based autonomous tool deployed with the UI-TARS 7B model (and tested with 2B SFT) to assess its capabilities for web navigation and financial data extraction.

Developed by ByteDance Research and hosted at Github link below, this agent was tested between February and March 2025 following Midscene's limitations.

https://github.com/vishwamartur/UI-TARS-desktop

Our team, Olwin Christian (HDFC Bank tests), Rahul Thakur (benchmarks), and Iremide Oloyede (scraping), explored its performance, runtime stability, and suitability for financial tasks, guided by mentors Feras and Kukesh, and contributions from other team members to refine the model's performance.

While UI-TARS showed promise, its complexity and resource demands shaped our final conclusions.

Technical Implementation

UI-TARS Desktop was installed as a standalone application on Windows/Linux via its GitHub releases page, configured with Hugging Face Endpoints provided by Mike Fuller:

- **Base URL** - https://pktofecpti9fyu52.us-east-1.aws.endpoints.huggingface.cloud/v1/
- **Token** - Standard Hugging Face authentication.
- **Models** - UI-TARS 7B DPO (primary), 2B SFT (secondary).
- **Cost** - $1.8/hour, scaling to zero after 15 minutes, occasionally yielding 503 errors.

Setup followed the repo's deployment guide:

1. **API Service** - Launched via python -m vllm.entrypoints.openai.api_server --served-model-name ui-tars --model <path>.
2. **Local Install** - Run pnpm install && pnpm run dev after downloading the 7B-DPO model.

Sample task prompt (Olwin's HDFC test)

```
{
  "task"  "Calculate HDFC Bank loan interest"
  "url"  "https://hdfc_bank_loans"
  "input"  "screenshot + rate details"
}
```

Rahul's Benchmarks

model = Qwen2VLForConditionalGeneration.from_pretrained("bytedance-research/UI-TARS-7B-DPO", torch_dtype=torch.float16, quantization_config=BitsAndBytesConfig(load_in_8bit=True))

tokenizer = AutoTokenizer.from_pretrained("bytedance-research/UI-TARS-7B-DPO")

Tests ran on Colab (GPU) and local Windows setups, with 2B facing frequent endpoint issues and 7B causing runtime instability.

Performance Evaluation - HDFC Bank EMI Extraction

Task: Scrape and calculate EMI from HDFC Bank's loan page using UI-TARS 7B DPO (Olwin).

- **Input** - Screenshot + prompt (~500 tokens).
- **Expected Output** - EMI for ₹315,000 at 12.5% over 5 years (~₹8,400).
- **Outcome** - Successfully computed ₹8,412 after ~2 minutes of loading and processing, surpassing Midscene's token-limited failures, but failed to screenshot and save the result to the desktop.

Iremide's 2B SFT scraping test *(Bankrate.com)*

model = AutoModelForVision2Seq.from_pretrained("bytedance-research/ui-tars-2b-sft").to("cuda")

inputs = tokenizer("Find mortgage rates on bankrate.com...", return_tensors="pt").to("cuda")

output = model.generate(**inputs, max_length=1500)

- Output: [{ "tool": "Click", "params": { "selector": "text=\"CD rates\"" } }]
- Result: Failed due to OCR errors and 503 timeouts; only 7B handled dynamic pages reliably.

Rahul's GSM8K benchmark (7B SFT)

Question **-** Janet's ducks lay 16 eggs daily...

Ground Truth **-** 18

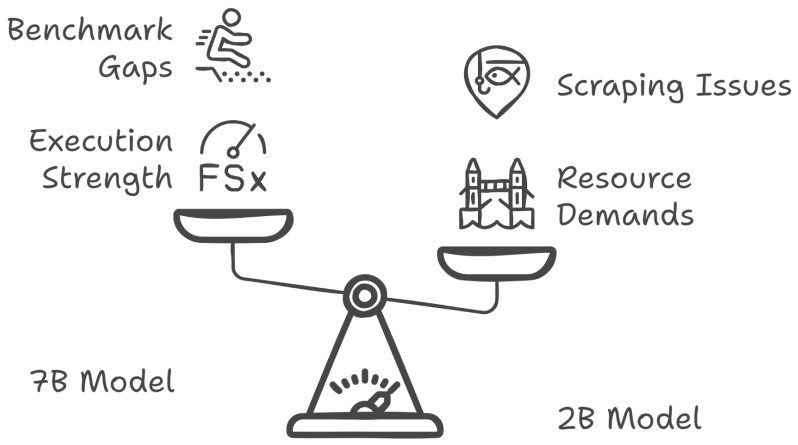
Prediction **-** 18

Accuracy **- 0.4** (2/5 correct)

Key Observations and Limitations

UI-TARS Desktop with 7B excelled in practical tasks but faltered in scalability:

- **Execution Strength** - Outperformed Midscene in EMI extraction, leveraging screenshot-based navigation and OS control.
- **Resource Demands** - 7B required GPU and endpoints, crashing Colab runtimes; 2B suffered 503 errors and poor reasoning (10% HellaSwag accuracy).
- **Benchmark Gaps** - Rahul's tests showed 40% GSM8K, 10% HellaSwag, and ROUGE-1 ~0.14 for summarization, weak without multimodal finetuning.
- **Scraping Issues** - Iremide's 2B OCR approach failed on dynamic content, echoing Scrum's prompting woes.
- **Complexity** - OS-level features exceeded our web-focused needs, per Kukesh's guidance toward lighter options.



Evaluating Model Performance and Resource Efficiency

Conclusion

The UI-TARS Desktop agent, deployed with the 7B DPO model, delivered impressive capabilities in web and GUI navigation, notably outperforming Midscene in Olwin's HDFC Bank EMI extraction test by efficiently handling multimodal inputs and dynamic page interactions.

Rahul's comprehensive benchmarks across SQuAD, GSM8K, HellaSwag, and CNN/DailyMail revealed its potential, achieving 40% accuracy on math reasoning tasks, yet exposed inconsistent reasoning (10% on HellaSwag) and modest summarization scores (ROUGE-1 ~0.14), suggesting limitations in broader applicability without extensive finetuning.

Iremide's scraping trials with the 2B SFT model further underscored inefficiencies, as OCR errors and endpoint timeouts (503 errors) hampered performance on sites like https://www.bankrate.com/ , while the 7B model, though more reliable, demanded significant GPU resources and occasionally failed secondary tasks like saving outputs to the desktop.

These findings highlighted a recurring theme: high computational overhead and complexity that often exceeded our project's web-focused requirements. Guided by mentor Kukesh's recommendation to prioritize efficiency and alignment with our core objectives, we concluded that while UI-TARS Desktop offered powerful tools for financial data extraction, its resource intensity and extraneous features prompted a strategic shift toward a leaner, more tailored solution like proxy-lite to better serve our streamlined, web-centric goals.