

WOS

Presented by:

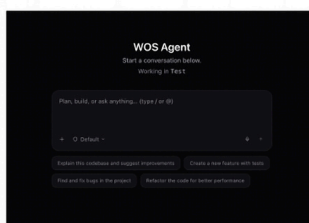
Jayam Shah
Thejes Raj Gangadhar
Yashwanth Reddy Katipally
Winston Chen

Project Advisors:

Dr. Simon Shim

Introduction

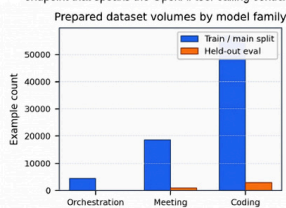
Most AI tools live in the cloud. WOS lives on your machine. WOS is a desktop AI agent that reads your files, runs code, connects to your apps, and remembers past conversations -- without sending any data to an external server. No subscriptions, no accounts, no data leaving your computer. It integrates with Slack, GitHub, Jira, and Google out of the box. You can extend it with plain-English Skills and Rules, or plug in any community tool server using the Model Context Protocol standard.



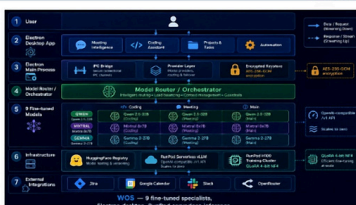
Methodology

We treated reliability as a data problem, not a prompt problem:

1. Generate. We synthesized multi-turn tool-use trajectories that cover Slack, GitHub, Jira, and Google, plus single-step, triage, and cross-app patterns. Errors and clarification turns were injected at calibrated rates so the model learns to recover.
2. Audit and repair. Each trajectory is checked against strict JSON schemas. Bad records are repaired or rejected, never patched silently.
3. Split. Train, validation, and blind-test sets are scenario-disjoint, so evaluation is honest.
4. Fine-tune. Qwen3-32B is trained with QLoRA in 4-bit, then served on a Hugging Face Space behind a vLLM endpoint that speaks the OpenAI tool-calling contract.

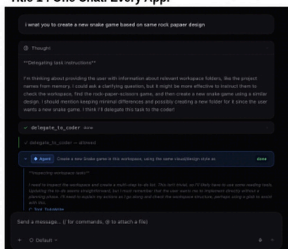


Analysis and Results



WOS is an Electron desktop app with a strict main and render split. The main process owns orchestration, tool execution, and an encrypted SQLite store. The render is a clean React surface for chat, projects, meetings, automations, and apps. A provider layer lets the same conversation run on OpenAI, Anthropic, or our own fine-tuned model with no code change in the UI.

Title 1: One Chat. Every App.



Structured eval, 27 scenarios across ten behavioral categories:

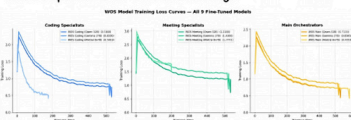
Model	Score	%	Halluc.	Last. gen (s)	Last. mem (s)
wos-vllm (fine-tuned Qwen3-32B)	27/27	100%	0	3.24	3.24
Qwen3-32B-bnb-4bit (base, think off)	22/27	81%	0	7.56	8.94
meta/Llama-3.3-70B-instruct (NIM)	24/27	89%	0	1.63	4.33

Blind chain eval, 5 multi-step workplace tasks:

WOS fine-tuned	80%
Llama 3.3-70B (NIM)	80%
Qwen3-32B base	0%

The fine-tuned 32B matches the 70B frontier model on real multi-step work, with zero hallucinations.

Title 2: Specialists for Code and Meetings



Coding specialist:
HumanEval pass@1 80.0% (matches an untuned 70B baseline)
MBPP pass@1 55.0% (matches the same baseline)

Meeting specialist on DialogSum:
ROUGE-1 F1 22.23 (baseline 19.77, +2.46)
ROUGE-2 F1 6.22 (baseline 5.65, +0.57)
ROUGE-L F1 16.65 (baseline 14.43, +2.22)

The meeting model writes summaries that recall more of the source. The coding model holds the line at a fraction of the size.

Title 3: How WOS Thinks

The agent runs a tight loop. Plan the next step. Pick a tool. Check the call against a JSON schema. Run it. Read the result. Decide if the task is done.



Every step is logged. Every tool call is typed. If a tool fails, the loop sees the error and tries a different path. The user always sees what was done and why.

Title 4: Works in the Wild

These five multi-step tasks were never seen during training. WOS ran each one end to end, choosing the right tools and the right order without guidance.

Task	WOS	NIM (70B)	Base
Incident response triage	100%	100%	0%
Sprint planning prep	83%	100%	0%
Weekly cross-app digest	83%	67%	0%
New feature kickoff	100%	100%	0%
Stale PR triage	33%	33%	0%
Average	80%	80%	0%

Each task is a 3 to 4 step chain across Slack, GitHub, and Jira. The base model scores zero on every chain. WOS and the 70B frontier model are neck and neck, at one-twelfth the size.

The hardest task (PR triage, 33%) tripped both WOS and the frontier model the same way: both retrieved the pull requests correctly, then asked for clarification instead of filing the Jira ticket. Same failure, same root cause, different scale.

Summary/Conclusions

WOS shows that a careful contract beats a bigger model. The same orchestration quality you would expect from a 70B hosted endpoint is delivered by a 32B model we trained, hosted on our own Space, at one-twelfth the parameter count.

What we built:

- A desktop assistant that unifies Chat, Projects, Meetings, Automations, and Apps.
- A 29-table SQLite store with AES-256-GCM encrypted credentials.
- A provider layer with OpenAI, Anthropic, and self-hosted vLLM, all behind one streaming contract.
- A fine-tuned orchestrator that scores 100% on the structured suite, 80% on blind chains, and 0 hallucinations.

What it means:

Reliable agentic work does not require a frontier price tag. With strict schemas, audited trajectories, and honest splits, smaller models can run production workflows.

Key References

- [1] Yao, S. et al. ReAct: Synergizing Reasoning and Acting in Language Models. ICLR, 2023.
- [2] Anthropic. Model Context Protocol Specification. modelcontextprotocol.io, 2024.
- [3] Hu, E. et al. LoRA: Low-Rank Adaptation of Large Language Models. ICLR, 2022.
- [4] Detrmers, T. et al. QLoRA: Efficient Finetuning of Quantized LLMs. NeurIPS, 2023.
- [5] Kwon, W. et al. Efficient Memory Management for Large Language Model Serving with PagedAttention (vLLM). SOSP, 2023.
- [6] Chen, M. et al. Evaluating Large Language Models Trained on Code (HumanEval). arXiv:2107.03374, 2021.

Acknowledgements

Thanks to RunPod for GPU infrastructure access, and to the Applied Data Science faculty for guidance throughout this project. We also thank the open-source teams behind Qwen, vLLM, and Hugging Face Spaces for the infrastructure that made our serving stack possible.

Interested in partnering with us on a capstone project or exploring other ways to collaborate with the College of IDS? Contact CIDS_CAP@sjsu.edu to discuss potential projects and partnership opportunities.