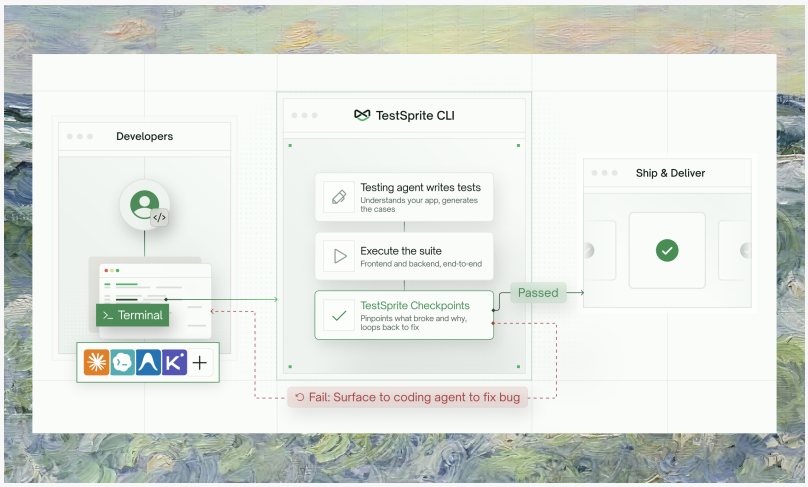
TestSprite Open Sources a CLI That Lets AI Coding Agents Autonomously Verify Their Own Work

Released under Apache 2.0, the CLI catches the bugs and regressions agents introduce coding unsupervised, with public proof live in CoderCup.



How the TestSprite CLI closes the loop: it tests an agent's work against the live app, pinpoints what broke and loops fixes back until the build passes.

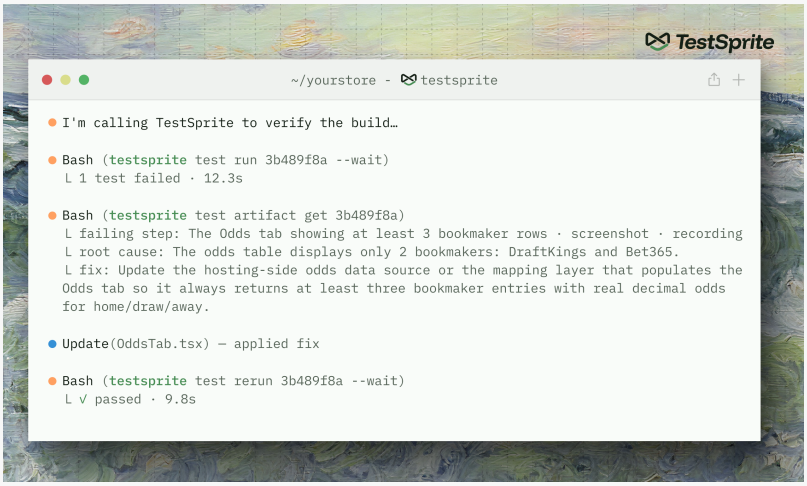
SEATTLE, WA, UNITED STATES, June 11, 2026 /EINPresswire.com/ -- TestSprite, the verification backbone for the agentic software era, today released the TestSprite CLI. It's the first open source command-line tool that lets AI coding agents autonomously verify their own work across both frontend and backend before declaring a task complete. Available under the Apache 2.0 license at github.com/TestSprite/testsprite-cli, it's purpose-built for the new generation of AI agents that now run for hours without a human present. The CLI is live and already proving itself in CoderCup, an open competition at codercup.ai where frontier AI coding agents, including Anthropic's Claude Code, OpenAI Codex, and Google Antigravity, each build the same real-world web app while TestSprite verifies their work.

> Even the best agent in our competition broke 12% of the features that already worked. That's the gap a verifier closes."
>
> *Yunhao Jiao*

The verification gap of the autonomous coding era

Over the past six months, AI coding agents crossed a threshold: the leading agents can now run unattended through the night, writing software while developers sleep. But building faster doesn't mean building right. An agent will routinely report a feature as "complete" when it has shipped a page that doesn't even render. Other times it will build a function that runs but silently breaks something else. The bottleneck of AI-native development has quietly moved from writing code to verifying it. But every way we verify code today, from IDE plugins to dashboards, was built for a developer sitting at a screen.

An agent working overnight isn't sitting at a screen; it lives in the terminal. So its verifier must live there too: a command the agent can run on its own, with no human in the loop.



A coding agent runs the TestSprite CLI, gets a failing test with the root cause and a suggested fix, applies it, and reruns to a clean pass, all from the terminal.

How CLI closes the loop
The CLI gives a coding agent a real QA loop, not a one-shot check. The agent describes a behavior once. TestSprite runs it in the cloud the way a real user would, driving a live browser or hitting a live API, never asserting on mocks. It returns a single, self-consistent failure bundle: the failing step and its neighbors, screenshots, DOM snapshots, the test source, a root-cause hypothesis, and a recommended fix, all tied to one snapshot. The agent reads the bundle, fixes the code, and reruns.

Every test that passes is kept, and that's where the real leverage builds. Each time the agent finishes a phase of work, TestSprite adds dozens of new tests, so coverage grows in lockstep with the codebase. Early on, when the app is small, the safety net barely matters.

Across a long, multi-phase build, however, it becomes the line between shipping and shipping broken. By the mid-to-late phases, the suite is large enough to catch the three failures that quietly sink autonomous work: real bugs in what the agent just wrote, features it reported as "done" that never actually worked, and regressions, where a later change silently breaks code that used to work. Regressions are the ones an agent can't catch on its own: it didn't think it touched that feature, so it never goes back to look. A suite that reruns everything, on every change, does.

New Metrics Revealed: The Verification Loop That Makes Software Self-Evolve
Running TestSprite across the CoderCup field has revealed something the industry has never systematically measured: how a coding agent behaves over a long build, not just whether a single task passes. TestSprite is publishing it as a new family of metrics that includes what an agent gets right on the first try, what it fixes after a failure is pointed out, what it never gets right, and what it breaks along the way. That last category, regressions, where a later change silently breaks working code, is the one no speed-or-cost benchmark has ever tracked, and the one developers feel most.

Two findings stand out. First, agents can self-evolve once you give them a verification loop: one agent began a phase of CoderCup with zero of its target features working. After ten rounds of reading TestSprite's failure feedback and fixing what it broke, the same agent and same underlying model, finished with roughly 80% of those features passing. The only thing that changed was that it finally had something to verify against. Second, regression is everywhere. The strongest run we've measured still broke roughly 12% of its previously passing features in a single run. Weaker runs approached 25%, the single biggest reason a developer still must babysit a so-called "autonomous" agent.

A growing test suite turns out to do more than catch bugs. It becomes a memory the model doesn't have. No context window, not even a million tokens, holds every requirement across a long build. So a failing test is what puts a dropped constraint back in front of the agent mid-task.

And over enough phases, it changes which model you even need. In CoderCup, smaller, cheaper models reached the same feature-completeness as frontier ones after a dozen-plus iterations, at a fraction of the total time and cost. That's because the verifier, not raw model strength, is doing the heavy lifting.

That's what TestSprite is putting in front of the industry: a shared vocabulary for how agents behave over time, and a path to production-quality software that doesn't always demand the biggest model.

"That's exactly what's driving developers crazy," said Yunhao Jiao, founder and CEO of TestSprite. "You use AI, you ship something new, you fix one thing and then boom, another thing crashes. Even the best agent in our competition broke 12% of the features that already worked. That's the gap a verifier closes. Until you can prove the work actually holds up, an autonomous coding agent is just a faster way to introduce instability."

The live proof: CoderCup

CoderCup, live at codercup.ai, is the public proof that the agent-to-verifier handshake works at scale. A growing field of frontier coding agents each build the same ten-phase application under identical rules and an identical clock. TestSprite serves as the neutral verifier and referee, scoring every phase against 16 to 22 end-to-end test plans. The full task specification, scoring rubric, per-phase results, and open repository are all published at codercup.ai. Anyone can clone it and re-run a phase to check the work.

"We're publishing these numbers because the industry has been graphing speed and cost while ignoring what developers live with every day," Yunhao added. "Regression and self-maintenance are how an autonomous coding agent actually gets judged, and CoderCup measures both, every phase, in public."

Availability

The TestSprite CLI is open source under the Apache 2.0 license and available today.
Install: npm install -g @testsprite/cli (requires Node.js 20 or higher)
Documentation & CLI reference: https://github.com/TestSprite/testsprite-cli
CoderCup repository: https://github.com/TestSprite/CoderCup

About TestSprite
TestSprite is the verification backbone for the agentic software era. Its testing engine generates fresh test plans, runs them against your live application the way a real user would, and returns machine-readable verdicts that AI coding agents and human engineers can act on directly, the foundation for self-evolving software, where agents iterate on and finalize their own code. The platform spans the full software lifecycle, from a developer's IDE to an AI coding agent's terminal to every pull request in CI/CD. The TestSprite CLI is open source under the Apache 2.0 license. Based in Seattle, TestSprite powers the workflows of more than 100,000 development and QA teams worldwide. Learn more at testsprite.com.

Editor's Note: Artwork available upon request.

Carmen Hughes
Ignite X
+1 650.576.6444
email us here