SwiftUI Agent Workflows Get Real as Cursor Pushes Demo-First Review

Coding Agents Alpha Tracker

2026-03-28

SwiftUI Agent Workflows Get Real as Cursor Pushes Demo-First Review

By Coding Agents Alpha Tracker • March 28, 2026

Simon Willison published the most copyable workflow of the day: small native SwiftUI tools built with Claude Opus 4.6 and GPT-5.4, Git checkpoints, and aggressive pattern reuse. Also inside: Cursor's million-commit signal, Codex starter prompts, Kody's secret-safe auth flow, and the eval checklist worth stealing.

TOP SIGNAL

Today's best practical signal is Simon Willison's fully documented SwiftUI loop: Claude Opus 4.6 and GPT-5.4 were good enough to build tiny macOS utilities as single-file apps without opening Xcode [1]. His repeatable pattern is simple: start from a concrete monitoring question, ask for a native app in `/tmp`, checkpoint early with `git init` and `git commit`, then add narrow follow-up prompts until the app becomes a menu bar tool [1]. The caveat is the useful part too: Willison says he barely reviewed the generated code and was not confident the reported system metrics were always correct, so treat this as a fast prototyping loop, not automatic ground truth [1].

TOOLS & MODELS

- **SwiftUI is suddenly a live agent target.** Simon says Claude Opus 4.6 and GPT-5.4 are both competent enough at SwiftUI to build single-file macOS apps without Xcode; separately, Codex now has a use-case gallery with starter prompts you can open directly in the app, including an iOS workflow with SwiftUI skills folded into the plugin [1, 2, 3].
- **Cursor is leaning into autonomy plus richer review.** Michael Truell says Cursor cloud agents produced over a million commits in the last

two weeks and that these were essentially all AI-generated with little human intervention because the agents run code on their own computers [4]. Cursor also now shows demos, not diffs: agents can use the software they build and send back video of the result [5].

- **Kody is shipping a real secret-boundary pattern.** Kent C. Dodds says Kody can now dynamically handle secrets for auth with any provider and any auth mechanism without giving the model access to the secret [6]. He used that flow from Claude to build a Spotify control app with secure OAuth handling [7].
- **Claude Code got a power-user hook filter.** Hooks now support an `if` field using permission-rule syntax, so you can run them on specific bash commands instead of every command [8].
- **Linear's agent numbers are getting hard to ignore.** Theo highlighted Linear's reported stats: coding agents are installed in more than 75% of enterprise workspaces, agent-completed work volume grew 5x in the last three months, and agents authored nearly 25% of new issues. Linear also rolled out a workspace agent, Skills, and Automations [9].

WORKFLOWS & TRICKS

- **Steal Simon's tiny-native-app loop.** Ask the agent the underlying diagnostic question first; once it proves the data is accessible, prompt it to create a native SwiftUI app in `/tmp`; immediately commit a working baseline; ask the model to suggest next features; then request one concrete improvement at a time such as per-process stats, reverse DNS, layout changes, and a menu bar icon [1]. For the next app, point the agent at the first repo and tell it to imitate the useful pieces [1]. Transcript + code: Bandwidther, Gpuer, repo 1, repo 2 [1].
- **Use Git as agent memory, not just backup.** Willison's explicit move was `git init and git commit what you have so far` before branching into new features, and he argues repositories are core tooling for ambitious agent work because they let you inspect and reverse changes cleanly [1, 10].
- **Teach the framework before asking for the app.** Simon had Claude clone Starlette 1.0 and generate a skill markdown with feature examples, copied that into Claude skills, then started a fresh session asking for a task app. Claude wrote the app and manually tested it with `TestClient`, which is a very usable pattern for framework-heavy work [10]. Repo: TaskFlow [10].
- **Keep secrets out of model context.** Kent's concrete prompt was `Use Kody to create an app that can control my Spotify`; Claude then guided app setup, handled OAuth securely, and never needed the tokens or client secrets in-model. He says the same pattern works in any MCP-supporting client [7, 11]. Demo link: Spotify control app [12].
- **Eval coding agents like software, not vibes.** LangChain's checklist is concrete: manually review 20-50 real traces before building eval in-

frastructure; define unambiguous pass/fail; separate capability evals from regression evals; start with full-turn evals that verify final response, trajectory, and actual state changes; for multi-turn, use N-1 testing; and run coding-agent trials in fresh containers or VMs before wiring regression evals into CI/CD gates [13]. Full checklist: LangChain's guide [14].

- **Tune the tool surface before the prompt.** LangChain's most reusable advice: tool interface design can eliminate entire classes of agent errors; their example is requiring absolute file paths so navigation mistakes become impossible [13].
- **Recurring pattern across operators: prototype first, spec second.** Theo says his old Twitch loop was to build a scrappy version in 1-3 days to test UX and surface technical constraints, then either write a much better spec or just ship the first pass. He still thinks code is often the best planning artifact [9].

PEOPLE TO WATCH

- **Simon Willison** — still one of the best public labs for agentic coding because he publishes the prompts, transcripts, repos, and the warnings about what he does *not* trust yet [1].
- **Kent C. Dodds** — worth following if you care about agent security that survives contact with real APIs; Kody's secret-handling and Spotify demo are concrete, not theoretical [6, 7, 11].
- **Romain Huet** — good pulse on where Codex is getting operationally better: starter-prompt workflows today, and strong user anecdotes on bug-fixing performance. In one example he amplified, a user said Codex one-shotted a bug that Claude Code had spent four hours on, and Huet said he sees many cases like that [2, 15, 16].
- **Michael Truell** — if Cursor's cloud agents are already pushing over a million essentially AI-generated commits in two weeks, that is a real scale signal for unattended code execution [4].
- **Theo** — strong signal on workflow judgment: prototype-first building, skepticism of role-played org-chart skills, and a useful reminder that automations may be more valuable than many devs assume [9].

WATCH & LISTEN

- **6:07-9:29 — Theo on prototype-first development.** Worth the time for one specific loop: build a rough version fast, use it to discover UX and dependency truth, then decide whether to spec properly or just ship the prototype [9].

Jira and Linear are legacy software (6:07)

- **17:06-20:31 — Theo on code as planning.** The spicy part is optional; the useful part is his claim that role-played agent personas and bloated planning docs are often worse than letting the model do a first-pass build and learning from that artifact [9].

Jira and Linear are legacy software (17:05)

PROJECTS & REPOS

- **Bandwidther** — network bandwidth monitor built from minimal prompts. The real value is that the full transcript is public, so you can copy the exact build loop instead of guessing [10].
- **Gpuer** — GPU/RAM monitor built by pointing the agent at Bandwidther and asking for a similar app. Best example in today's batch of repo-to-repo recombination as a workflow primitive [10].
- **TaskFlow** — useful less as an app and more as a pattern: create a framework skill first, then generate and test a fuller app against that skill [10].
- **Kody PR #61** — secure secret routing for any provider and auth mechanism without exposing the secret to the model. This is one of the most important practical safety patterns in today's feed [6, 17].

Editorial take: the sharpest teams today are not chasing maximum autonomy; they're building small reusable loops, keeping Git and evals close, and drawing hard boundaries around secrets and real-world state [1, 10, 13, 6].

Sources

1. Vibe coding SwiftUI apps is a lot of fun

2. X post by @romainhuet
3. X post by @romainhuet
4. X post by @mntruell
5. X post by @cursor_ai
6. X post by @kentcdodds
7. X post by @kentcdodds
8. X post by @lydiahallie
9. Jira and Linear are legacy software
10. Vibe coding SwiftUI apps is a lot of fun
11. X post by @kentcdodds
12. X post by @kentcdodds
13. Agent Evaluation Readiness Checklist
14. X post by @LangChain
15. X post by @VadimStrizheus
16. X post by @romainhuet
17. X post by @kentcdodds