Persistent Specs and Auditable Outputs for Coding Agents

Coding Agents Alpha Tracker

2026-07-27

Persistent Specs and Auditable Outputs for Coding Agents

By Coding Agents Alpha Tracker • July 27, 2026

Persistent project context is the day's strongest coding-agent pattern: use agent-readable design rules to prevent UI drift, then apply the same auditability to financial models and task planning. Also covered: a cleanup pass for over-eager code changes and Claude Code/Codex packaging differences.

TOP SIGNAL

Give the agent a persistent design spec, not just tokens. AI For Developers frames inconsistent agent-generated UI across sessions as a context problem: without retained visual rules, agents fall back to generic defaults. A root-level `DESIGN.md`, explicitly loaded from `CLAUDE.md`, gives the agent reusable rules for visual roles, component states, layout, and prohibited patterns. [1]

TRY THIS

- **Add durable visual memory to a coding-agent repo.** Create `DESIGN.md` at the project root, then add a visual-rules section to `CLAUDE.md` that requires the agent to read and follow it before generating UI. Define color *roles*, typography hierarchy, component states, spacing, elevation, responsive behavior, and explicit do/don't rules—not merely hex values and font names. [1]

  ```
  Follow the rules defined in @DESIGN.md strictly for
  all UI generation.
  ```
  [1]

  A useful constraint style is semantic: "brand-primary is for CTAs and links only—never backgrounds, never dividers." [1] Start by asking for one

button and one card before handing the agent a full screen; this exposes underspecified rules early. [1]

- **Make research-generated spreadsheets auditable.** Riley Brown shared a Claude Excel + Opus 5 workflow: constrain web research to named primary sources, then require a Sources tab with URLs, filing dates, reporting periods, page numbers, and retrieval dates. Separate assumptions from three years of historical statements; build forecasts with live formulas, supporting schedules, a DCF sensitivity table, checks, and an executive dashboard. [2]

  ```
  Never invent a missing figure-mark it as unavailable
  and explain the gap.
  ```
  [2]

  Brown reports that this single-prompt NVIDIA analysis ran research for 37 minutes and generated a linked workbook. [3, 2]

- **Turn brainstorms into an executable backlog.** After a planning conversation produces markdown notes, send the conversation to Linear through its MCP integration and have it generate the actionable ticket set. For daily prioritization, ask the agent: "go to LINEAR and tell me what I need to do next." ThePrimeTime's host describes using this to convert ideas into tickets and choose the next task. [4]

- **Add a scope-trimming pass after high-initiative coding.** Theo reports that Claude Opus can notice adjacent issues and add roughly 100 lines of fixes plus 200 lines of tests; he often brings in Fable to trim the resulting excess. Treat the cleanup pass as a separate review step: retain only changes necessary to the requested outcome. [5]

WHAT SHIPPED

- **Claude Excel extension: practical modeling signal.** Riley Brown is testing Claude's Excel extension with Opus 5 and shared a detailed, source-constrained financial-model prompt—not a generic spreadsheet request. The notable pattern is the required provenance, formula-only derived values, visible input/formula formatting, and explicit data-gap warnings. [3, 2]

- **Claude Code vs. Codex: different subscription trade-offs.** Theo notes both offer fast mode and support context windows up to 1M tokens on their best models. His comparison: Claude Code includes 1M context in the base subscription but charges extra for fast mode; Codex includes fast mode but charges extra for higher context windows. He considers the distinction interesting but not especially consequential. [6]

GO DEEPER

- **02:54–03:10 — Linear MCP as a "what next?" layer.** A short workflow demo: turn a large planning conversation into tickets, then let the agent query Linear when you need the next concrete task. [4]

Open Sauce 2026 | TheStandup (2:54)

- **Repo to study: awesome-design-md.** This community library collects pre-written `DESIGN.md` files modeled on products including Linear, Stripe, Vercel, and Notion, with HTML previews. Use it as a reference for the sections and specificity an agent-readable visual spec needs. [1]

Editorial take: the strongest agent workflows make constraints durable and inspectable—whether the artifact is a UI spec, a cited spreadsheet model, or a ticket backlog. [1, 2, 4]

Sources

1. DESIGN.md for Design in Claude Code
2. X post by @rileybrown
3. X post by @rileybrown
4. Open Sauce 2026 | TheStandup
5. X post by @theo
6. X post by @theo