

AI Job Tracker

Examples from two autonomous scheduled agents built in Cowork

Craig Stevenson | mysorethumbs.com | Generative AI project 3 of 3

Overview

I wrote the instructions for two autonomous scheduled agents that search job boards, filter and score postings against a written profile, and append new matches to a spreadsheet tracker. A script regenerates an HTML dashboard after every run. A separate panel of review subagents helps me decide which postings are worth pursuing. Every new row lands as **To review**, so the agents assist my decisions and never make them.

System at a glance	
Agents	Two scheduled tasks. Each run covers two tracks: core game design, and immersive training / instructional design.
Tools	Web fetch for static sites, browser automation for client-rendered sites, LibreOffice and openpyxl for the spreadsheet, a Python dashboard script.
Outputs	A tracker spreadsheet (Applications and Rejected sheets) and an HTML dashboard that lists "To review" rows first.
My role	Wrote the agent instructions, defined the match rules, scoring rubric, and safety checks, and debugged failures as they appeared.

The two agents

Agent	Schedule	Method	Sources (examples)
Daily scan	Every day, 8:05 AM	Static web fetch only, no browser.	Design-focused and remote game job boards; an EdTech job board searched with four keyword queries (instructional design, learning experience designer, game design, UX design).
Browser scan	Monday and Friday, 9:00 AM	Browser automation, because these boards need JavaScript rendering or interactive filters.	Hitmarker, Game Jobs Direct, GameJobs.co, ASGC (a games job aggregator), HealthySimulation.com, Teamed for Learning.

Splitting the work this way keeps the cheap, reliable fetches on a daily schedule and reserves the slower, more fragile browser work for twice a week.

Example 1: Protecting the data

The agents edit a spreadsheet I also edit by hand, so most of the instructions exist to make sure an unattended run can never damage it.

Always start from fresh state

Convert a FRESH copy to xlsx for editing purposes only. Always regenerate this conversion at the start of the run. Never reuse a leftover scratch file from a previous run, since that would contain stale data and clobber manual edits.

Add and reorder only. then verifv before saving

Before making any changes, snapshot every existing data row as a full tuple of its column values. Safety check before finalizing: every snapshotted row must still exist somewhere in the final sheet with all its values unchanged. You are only ever allowed to ADD new rows and REORDER existing ones, never edit or delete the contents of an existing row. If any snapshotted row is missing or altered, STOP, do not save, and report the discrepancy in the summary instead.

Two-stage duplicate detection

Primary match: normalized job posting URL equals any existing row's URL.
Fallback match: normalized company + role title equals any existing row, checked even when both postings have URLs, since the same role is sometimes re-listed under a new URL. If either match hits, skip it and do not modify the existing row. If the URL differs, skip it and note the discrepancy in the summary.

Example 2: Degrading gracefully

Job sites change, block requests, and render content in the browser. The instructions say what to do when each of those things happens instead of letting a run fail.

- **Browser tool unavailable:** skip that source for the run, state it clearly in the summary, and tell me to check that the browser extension is installed and signed in.
- **Silent redirects:** one board's plain fetch quietly returns an unfiltered page. The instructions note that it only works through real browser navigation and route it to the browser agent.
- **Filters that must be applied:** for the aggregator, the instructions give the exact steps (locate the Category control, set it, locate the Location Type control, set it) and require checking for the "Active filters" text and a filtered count before reading any results.
- **Rate limits:** one source kept returning HTTP 429, so it was replaced with a server-rendered alternative, and the instructions record why.
- **Tidy exit:** delete lock files and scratch files, then confirm that only the three expected files remain.

Example 3: Bugs the instructions now guard against

Several failures kept coming back until the instructions made the fix unconditional.

The header row that kept turning unreadable

After some runs the header lost its color and became black text on a black fill, and the damage returned after the tasks were updated. The fix is to reassert the header formatting on every run, with literal values and after the re-sort so row-moving logic cannot overwrite it.

Explicitly reassert the header row formatting unconditionally every run. Use explicit literal values (never a theme color reference, which is the likely source of the original corruption). Do this AFTER the re-sort so it can never be accidentally overwritten by row-moving logic. Spot-check the header's fill and font color before finalizing.

The frozen header that kept disappearing

The ods to xlsx conversion silently drops the freeze-pane setting on every round-trip, so set `ws.freeze_panes = "A2"` every run, unconditionally.

An outage outside the agents' control

A Windows update briefly broke the sandbox the tasks depend on for file access. Because the instructions already stop and report instead of saving partial work, the outage cost me a run rather than any data.

Example 4: Match rules and scoring

Each track has its own sites, profile, and criteria, and the instructions tell the agent not to mix them. The rules are written as hard filters plus judgment calls.

- **Hard filters:** work location and eligibility, role discipline, seniority range, and out-of-scope role types are excluded outright.
- **Exceptions are spelled out:** a requirement such as a specific engine excludes a posting unless it is listed as a plus or the posting says it will train.
- **Ambiguity is flagged, not hidden:** unclear postings are included, noted as ambiguous, and scored lower.

- **Weighted factors:** location fit, title and seniority fit, discipline fit, studio and genre fit, and tool fit, with core tools weighted over supporting ones.

Assign a Fit Score from 1-10 and a one-line Score Reason. Be honest and varied with scores. Do not cluster everything at 7-8. A genuinely excellent match should score 8-10. A plausible-but-stretchy match should land 4-6. Use scores 1-3 sparingly.

Example 5: The output row

Each new posting becomes one row with a fixed schema. The values below are placeholders for illustration, not real tracker data.

Field	Illustrative value
Company / Role	Example Studio (placeholder) / Senior Systems Designer
Status	To review
Fit Score	7
Score Reason	Remote senior systems role at an indie studio, Unity-based, no restrictive requirements.
Date Added	MM/DD/YYYY (the run date)
Job Posting	Plain URL, hyperlinked on that cell only
Location / Work Type	As listed in the posting
Salary Range	As listed, or "Not listed"
Notes	Why it matches, and which track (A or B) found it
Post Age	New
Panel Score / Verdict / Notes	Blank on purpose. Filled in later by the manual review panel.

The review panel

Scores from the agents are a first pass. For postings I am seriously weighing, I run a panel of subagents that reviews the role from three perspectives: a design lead, a hiring manager, and a recruiter. Their combined score, verdict, and notes are logged in the Panel Score, Panel Verdict, and Panel Notes columns. The panel runs on demand and is deliberately kept out of the scheduled tasks, which keeps the unattended runs simple and cheap.

What I learned

- **Guardrails come first.** The dedupe check, the verify-before-save step, and the stop-and-report rule matter more than how clever any single step is.
- **Plan for the environment to break.** Outages, rate limits, and client-side rendering are normal, so the instructions must say what to do when they happen.
- **Make recurring fixes unconditional.** When a bug kept returning, the lasting fix was to reassert the correct state every run instead of testing for the problem.
- **Several viewpoints beat one score.** Reviewing each posting as a design lead, a hiring manager, and a recruiter surfaces things a single read would miss.