Agent Mission Control — Claude Code Session Prompts

Track All Your Agents, Projects, and Progress in One Place

You're running multiple Claude Code agents across multiple projects simultaneously. You need a single screen that shows what's running, what finished, what broke, and texts you when something needs attention. This is your personal DevOps dashboard.

Why this could be a business:

- Every Claude Code power user has this exact problem — no visibility into agent progress
- The "AI agent orchestration" space is brand new — no dominant tool yet
- Solo devs and small teams running AI agents across repos need exactly this
- Free now, charge later when the feature set justifies it
- The indie hacker / AI builder community (Indie Hackers, r/SaaS, r/ChatGPT, Twitter/X builders) would eat this up
- Natural expansion: support other AI coding agents (Cursor, Copilot Workspace, Devin, etc.)

Architecture:

```
Your PC (Claude Code terminals)
  ↓ (lightweight watcher scripts push status)
Local Server (Python FastAPI on localhost)
  ↓ (serves dashboard + sends alerts)
Web Dashboard (Next.js — runs locally or deployed)
  ↓ (alert triggers)
Twilio / Pushover / ntfy.sh (text/push notifications to your phone)
```

SESSION 1: Status Watcher & Local Server

```
Build the core system that watches Claude Code sessions and collects status data.

**1. Session Watcher Script (Python):**
Create a lightweight Python script that monitors Claude Code activity on this machine.

a. **Terminal/Process Detection:**
  - Detect running Claude Code processes (look for claude, claude-code, or node processes related to Claude Code CLI)
  - For each detected session:
    - PID
```

- Working directory (which project repo)
- Start time
- Duration running
- Whether it's actively generating output or idle

b. **Log File Monitoring:**
- Claude Code writes logs/output to the terminal
- Watch stdout/stderr of each session using a log capture approach:
- Option 1: If Claude Code writes to a log file, tail that file
- Option 2: Use a wrapper script that pipes Claude Code output through tee to a log file
- Option 3: Monitor the terminal output via a tmux/screen session capture
- Parse the output for key events:
- "File created: [path]" → log as FILE_CREATED
- "Error" / "failed" / "TypeError" / "SyntaxError" → log as ERROR
- "Test passed" / "Tests: X passed" → log as TEST_RESULT
- "Build successful" / "compiled" → log as BUILD_SUCCESS
- "Build failed" → log as BUILD_FAIL
- "Deployed" / "live at" → log as DEPLOYED
- Session goes quiet for 5+ minutes → log as IDLE (might be done or stuck)
- Session ends → log as SESSION_COMPLETE

c. **STATUS.md Watcher:**
- Watch the STATUS.md file in each project repo for changes
- When it updates, parse the new content and extract:
- Current session number
- What's working
- What's broken
- Last session summary
- This is the bridge between Claude Code and our dashboard

d. **PROJECTS.md Watcher:**
- Watch your master PROJECTS.md file
- Parse all project statuses
- Detect when session statuses change (→ →)

2. Local API Server (Python FastAPI):
Create a FastAPI server running on localhost:9000

Endpoints:
- GET /api/sessions — list all detected Claude Code sessions with status
- GET /api/projects — parsed PROJECTS.md data as JSON
- GET /api/project/{name}/status — parsed STATUS.md for a specific project
- GET /api/events — recent events log (last 100 events)
- GET /api/events/stream — SSE (Server-Sent Events) endpoint for real-time updates

- POST /api/alert/test — send a test alert to your phone
- GET /api/health — server health check

Data model:

```
{
  "sessions": [
    {
      "id": "uuid",
      "project": "ai-chess-coach",
      "repo_path": "C:\\Users\\Kruz\\Desktop\\Projects\\idea3",
      "pid": 12345,
      "status": "running" | "idle" | "complete" | "error",
      "started_at": "2026-02-11T22:30:00",
      "duration_minutes": 45,
      "files_created": 12,
      "files_modified": 8,
      "errors_detected": 0,
      "last_activity": "2026-02-11T23:15:00",
      "last_output_snippet": "Created component PuzzleBoard.tsx"
    }
  ],
  "events": [
    {
      "timestamp": "2026-02-11T23:15:00",
      "project": "ai-chess-coach",
      "type": "FILE_CREATED",
      "message": "Created PuzzleBoard.tsx",
      "severity": "info"
    }
  ]
}
```

3. Launcher Script:
Create a simple launcher: start.bat (Windows) that:
- Starts the FastAPI server
- Starts the file watchers
- Opens the dashboard in the browser
- One double-click to launch everything

User runs: start.bat → dashboard opens → ready to monitor

TESTING:
- [ ] Watcher detects a running Claude Code process (start a Claude Code session, verify it shows up)
- [ ] STATUS.md changes are detected within 5 seconds

- [ ] PROJECTS.md changes are detected within 5 seconds
- [ ] FastAPI server starts on localhost:9000
- [ ] GET /api/sessions returns detected sessions
- [ ] GET /api/projects returns parsed project data
- [ ] GET /api/events returns recent events
- [ ] SSE endpoint streams events in real-time (open in browser, trigger a file change, see the event)
- [ ] Server handles no active sessions gracefully (shows empty state, doesn't crash)
- [ ] start.bat launches everything with one click

Show me the API responses with real data from your machine.

SESSION 2: Web Dashboard

Build the monitoring dashboard that shows all agent sessions and project status at a glance. This should be the one tab you keep open while agents run.

1. Project Setup:
- Next.js 14 App Router, TypeScript, Tailwind CSS, shadcn/ui
- Dark theme (you're staring at this while coding — needs to be easy on the eyes)
- Color palette: dark background (#0A0A0F), muted borders, with status colors:
  - Running/active: blue pulse (#3B82F6)
  - Success/complete: green (#10B981)
  - Warning/idle: amber (#F59E0B)
  - Error: red (#EF4444)
  - Not started: gray (#6B7280)
- The vibe: think Datadog or Grafana but personal and minimal

2. Main Dashboard Layout (/)

Top Bar:
- "Mission Control" title
- Clock showing current time
- Global status indicator: "3 agents running, 1 idle, 0 errors" with colored dots
- "All Systems Normal" / "⚠ Attention Needed" banner based on error state

Section 1: Active Sessions (hero section)
One card per running Claude Code session:

Each card shows:
- Project name + icon (chess piece for chess, chart for finance, etc.)
- Status badge: 🟢 Running / 🟡 Idle / 🔴 Error / ✅ Complete

- Runtime: "Running for 23 min"
- Current activity: last output line from the agent (updates in real-time via SSE)
- Progress indicator: "Session 5 of 10" with progress bar
- Mini stats: files created, files modified, errors
- Pulse animation on the card border when the agent is actively outputting (alive indicator)
- Click card to expand and see the full event log for that session

If no sessions are running: show "No active agents. Start a Claude Code session to begin monitoring."

Section 2: Project Overview (below active sessions)
One row per project from PROJECTS.md:

Project	Status	Progress	Last Activity	Next Step	Health
AI Chess Coach	🔄 Session 3	▓▓▓▓▓▓▓▓▓▓ 3/10	12 min ago	Playable game	🟢
AI Finance Brief	🔄 Session 6	▓▓▓▓▓▓▓▓▓▓ 6/8	2 min ago	Content depth	🟡
Trade Journal	⏸ Sidelined	▓▓▓▓▓▓▓▓▓▓ 2/7	2 days ago	—	🔵
PC Analyzer	⬜ Not started	▓▓▓▓▓▓▓▓▓▓ 0/6	Never	Python scanner	🔵

Click any row to see that project's full STATUS.md rendered nicely.

Section 3: Event Feed (right sidebar or bottom)
Real-time scrolling feed of all events across all projects:
- Timestamp + project badge + event message
- Color-coded by severity (info=gray, success=green, warning=amber, error=red)
- Filter buttons: All / Errors Only / Completions Only
- Search within events
- Auto-scrolls to newest, but stops auto-scroll if user scrolls up manually

Section 4: Stats Bar (bottom of page)
- Today's totals: sessions run, files created, errors fixed, tokens burned (estimated)
- This week's totals
- Streak: "5 consecutive days with coding sessions"

3. Project Detail Page (/project/[name]):
When you click a project from the overview:
- Full STATUS.md rendered as a clean page
- Session history table with status icons
- Event log filtered to this project
- Architecture decisions log
- Blockers highlighted prominently
- "Open in Claude Code" button (opens terminal to that directory — use shell command)
- Link to the session prompt doc for the next session

4. Real-Time Updates:
- Connect to FastAPI SSE endpoint on page load
- Dashboard updates WITHOUT page refresh
- New events slide into the feed
- Session cards update status in real-time
- Use React state management (useState/useReducer) — no complex state library needed

5. Responsive:
- Desktop: full dashboard layout (sidebar + main area)
- Mobile: stacked cards, collapsible sections (you might check this on your phone)

TESTING:
- [ ] Dashboard loads and connects to the FastAPI backend
- [ ] Active sessions show with real data from running agents
- [ ] Real-time updates work: start a Claude Code session → card appears on dashboard within 10 seconds
- [ ] Event feed scrolls and updates live
- [ ] Project overview shows correct progress for all projects
- [ ] Click a project → detail page shows STATUS.md content
- [ ] Error state: break something in a Claude Code session → dashboard shows error indicator
- [ ] Session completion: finish a Claude Code session → status changes to ✅
- [ ] No active sessions: empty state is clean and helpful
- [ ] Mobile: dashboard is usable on phone
- [ ] Page doesn't memory-leak after 30+ minutes of SSE streaming (check browser dev tools)

Show me the live dashboard monitoring at least one active session.

SESSION 3: Alert System (Text Notifications)

Build the alert system that texts you or sends push notifications when something important happens. This is what lets you walk away from the computer while agents run.

1. Notification Providers (implement at least 2):

a. **ntfy.sh (FREE — easiest, start here):**
  - Free, no account needed, open source
  - Send: POST https://ntfy.sh/your-secret-topic-name with a message body
  - Receive: install ntfy app on your phone, subscribe to your topic
  - No API key needed — just pick a unique topic name that nobody would guess
  - Example:
```python
  import requests
```

```
    requests.post("https://ntfy.sh/kruz-agent-alerts-x7k9",
        data="✅ Chess Coach Session 3 complete — 12 files created, 0 errors",
        headers={"Title": "Agent Complete", "Priority": "default", "Tags": "white_check_mark"})
```

b. **Twilio SMS (paid but reliable — add later):**
   - Send actual SMS texts to your phone number
   - Free trial gives you $15 of credits (~1000 texts)
   - Create account → get Account SID + Auth Token + Twilio phone number
   - ```python
     from twilio.rest import Client
     client = Client(account_sid, auth_token)
     client.messages.create(body="✅ Agent complete", from_="+1twilio", to="+1yourphone")

c. **Pushover ($5 one-time — great middle ground):**

- Push notifications to phone
- One-time $5 purchase for the app
- Rich notifications with titles, priorities, sounds
- Good for if you want different sounds for errors vs completions

2. Alert Rules: Create a configurable alert system. Default rules:

Event	Alert?	Priority	Message
Session complete (success)	✅ Yes	Normal	"✅ [Project] Session [X] complete — [files] files, [errors] errors"
Session complete (with errors)	✅ Yes	High	"⚠️ [Project] Session [X] done with [N] errors — check dashboard"
Build failure	✅ Yes	Urgent	"🔴 [Project] Build failed: [error snippet]"
Tests failed	✅ Yes	High	"❌ [Project] Tests: [X] passed, [Y] failed"
All tests passed	✅ Yes	Normal	"✅ [Project] All [X] tests passed"
Session idle >10 min	✅ Yes	Normal	"💤 [Project] Agent idle for 10+ min — may be stuck or finished"
Deploy successful	✅ Yes	Normal	"🚀 [Project] Deployed successfully"
Error spike (3+ errors in 5 min)	✅ Yes	Urgent	"🔴 [Project] Multiple errors detected — agent may be in a loop"
STATUS.md updated	❌ No	—	(Too noisy — just update the dashboard)

Store rules in a config file: alert_rules.json
Allow enabling/disabling each rule from the dashboard settings.

3. Alert Settings Page (/settings/alerts):

- Toggle each alert rule on/off
- Set notification provider (ntfy / Twilio / Pushover)
- Set phone number (for Twilio)
- Set ntfy topic name
- "Send Test Alert" button — sends a test notification to verify setup
- Quiet hours: don't send alerts between [time] and [time] (e.g., 11pm-7am)
- Alert cooldown: don't send more than 1 alert per project per 5 minutes (prevents spam during error loops)

4. Alert History Page (/alerts):

- Table of all alerts sent: | Time | Project | Type | Message | Delivered? |
- Filter by project, type, date range
- "Resend" button for failed deliveries
- Daily summary: "Today: 5 alerts sent (3 completions, 1 error, 1 idle warning)"

5. Daily Digest (optional but awesome): Every evening at 9pm, send ONE summary text:

"📊 Daily Recap:

- Chess Coach: Session 3 complete ✅, Session 4 complete ✅
- Finance Brief: Session 6 in progress 🔄
- 24 files created, 3 errors fixed
- Streak: 6 days 🔥 "

This is a single daily notification that summarizes everything even if you weren't watching the dashboard.

6. Integration with Dashboard:

- Dashboard shows a 🔔 notification bell icon
- Bell shows unread alert count
- Click bell → dropdown with recent alerts
- Alerts that triggered a text show a 📱 icon

TESTING:

- [ ] ntfy.sh setup: send a test alert → receive it on your phone (install the ntfy app first)
- [ ] Session complete alert: finish a Claude Code session → text arrives within 30 seconds
- [ ] Error alert: trigger an error in Claude Code → alert fires
- [ ] Idle alert: leave a session idle for 10 minutes → alert fires
- [ ] Quiet hours: set quiet hours to now → alerts don't send → unset → alerts resume
- [ ] Alert cooldown: trigger 5 errors in 1 minute → only 1 alert sent (not 5)
- [ ] Alert history page shows all sent alerts
- [ ] Settings page saves provider/config correctly
- [ ] Dashboard notification bell shows correct count
- [ ] Daily digest sends at the scheduled time with accurate data
- [ ] Twilio (if set up): test SMS delivery to your phone number

Show me a screenshot of a real alert on your phone.

SESSION 4: Claude.ai Integration & AI Insights

Connect the dashboard to Claude AI for intelligent status summaries and proactive suggestions. Also build the bridge between this dashboard and our Claude.ai Project conversations.

1. AI Status Summary: Add an "AI Summary" card to the top of the dashboard.

Every 30 minutes (or on demand), send all current data to Claude API:

"You are a project manager AI assistant monitoring a developer's active coding agents. Here is the current state of all projects and sessions:

[Full PROJECTS.md content]
[All active session data with recent events]
[Today's event log]

Provide a brief status update (5 sentences max):

1. What's happening right now across all projects
2. Any concerns (errors, stuck agents, sessions running too long)
3. What's the highest priority thing to do next
4. Any efficiency observations (e.g., 'Finance Brief is 75% done — consider pushing it to completion before switching to Chess Coach')
5. Motivational note about progress made today"

Display as a card at the top of the dashboard. Refreshes every 30 minutes or when user clicks "Refresh Summary."

2. Smart Recommendations: After each session completes, Claude analyzes the results:

"Session just completed for [project].
Here's what happened: [session events summary]
Here's the current STATUS.md: [content]
Here's the session prompt they were following: [reference to which session prompt doc]

Provide:

1. Was this session successful? (Yes/Partially/No)
2. Based on the STATUS.md, what should the next session focus on?
3. Any red flags in the session output (recurring errors, incomplete features)?
4. Is this project on track for launch? If behind, what's the fastest path to caught up?"

Show recommendations in a panel on the project detail page.

3. Context Export for Claude.ai: Build a "Sync to Claude.ai" feature:

- Button on the dashboard: "Generate Context Update"
- Produces a formatted text block optimized for pasting into a Claude.ai conversation:

```
=== AGENT STATUS UPDATE — [timestamp] ===

ACTIVE RIGHT NOW:
• [Project]: Session X running for Y min — last activity: [snippet]

COMPLETED TODAY:
• [Project]: Session X finished — [X] files, [X] errors — [summary]

PROJECT PROGRESS:
• Chess Coach: 3/10 sessions
• Finance Brief: 6/8 sessions
• Trade Journal: 2/7 sessions (sidelined)
• PC Analyzer: 0/6 sessions (not started)

BLOCKERS:
• Finance Brief: Resend API key needed for email testing

NEXT PRIORITIES:
1. [AI-recommended next step]
2. [AI-recommended next step]
```

- "Copy to Clipboard" button for easy paste
- This is the one thing you paste into our conversations instead of the full PROJECTS.md — it's denser and more current

4. Auto-Update PROJECTS.md:

- When a session completes and the STATUS.md updates, automatically update PROJECTS.md too
- Change the session status from 🔄 to ✅
- Update the date
- Update blockers if any were resolved
- This keeps PROJECTS.md always current without manual editing

5. Conversation Log:

- Store a log of all AI summaries and recommendations
- Accessible at /insights
- Over time this becomes a record of your development journey
- Searchable: "show me all recommendations about the chess coach project"

TESTING:

- [ ] AI summary generates with real data from your projects
- [ ] Summary updates every 30 minutes without manual refresh
- [ ] Smart recommendations appear after a session completes
- [ ] "Generate Context Update" produces a clean, pasteable text block
- [ ] Context update includes accurate session counts and progress bars
- [ ] Auto-update PROJECTS.md: complete a session → PROJECTS.md reflects the change
- [ ] Insights page shows historical AI summaries
- [ ] AI doesn't hallucinate projects or sessions that don't exist
- [ ] Works with 0 active sessions (shows "no active sessions" summary)
- [ ] Works with 4+ active sessions simultaneously

Show me the AI summary card with real project data.

SESSION 5: Session Launcher & Prompt Manager

Build a prompt management system that stores all your Claude Code session prompts and lets you launch sessions from the dashboard.

1. Prompt Library (/prompts):

- Store all session prompts from your prompt docs:
  - Chess Coach sessions 3-10
  - Finance Brief sessions 1-8
  - PC Analyzer sessions 1-6
  - Token maximization prompts
  - Post-MVP optimization prompts (14 prompts)
- Display as a card grid, organized by project
- Each card shows: prompt name, project, estimated duration, estimated token burn, status (done/next/later)

- Click to view the full prompt text
- "Copy to Clipboard" button on each prompt

2. Session Queue:

- Drag prompts into a "Queue" — ordered list of what you plan to run today
- Queue shows: estimated total time, estimated total token burn
- "Next Up" indicator on the dashboard highlights the next prompt in the queue
- When a session completes → automatically suggest the next queued prompt

3. Quick Launch:

- "Launch Session" button on each prompt card
- Opens a terminal window to the correct project directory
- Copies the prompt to clipboard
- Shows a toast: "Prompt copied! Paste into Claude Code to start."
- (We can't programmatically paste into Claude Code, but we can get you 90% of the way there)

Alternative if we can automate more:

- Generate a shell script per prompt that:
  - cd's to the project directory
  - Opens Claude Code
  - Puts the prompt in a file the user can cat and paste
  - start.bat equivalent: `cd C:\Users\Kruz\Desktop\Projects\idea3 && claude`

4. Prompt Editor:

- Edit prompts directly in the dashboard
- Markdown editor with preview
- Save changes (persisted to local JSON file or database)
- Version history: see past edits to a prompt
- "Duplicate and Modify" — fork a prompt to customize it

5. Prompt Analytics: After sessions complete, link results back to prompts:

- Which prompt was used
- How long the session took
- How many files were created/modified

- How many errors occurred
- Success rate per prompt
- "This prompt has been run 3 times with 67% success rate"

Show on each prompt card: "Last run: Feb 11, took 34 min, ✅ successful"

6. Template Prompts: Pre-built generic prompts that work on any project:

- "Full Code Audit" — paste the 1A prompt
- "Write Test Suite" — paste the 1B prompt
- "Generate Documentation" — paste the 2A prompt
- "Performance Optimization" — paste the 3A prompt
- User can apply any template prompt to any project

TESTING:

- [ ] All prompts from all prompt docs are loaded into the library
- [ ] Prompts are organized by project and show correct metadata
- [ ] Copy to clipboard works on every prompt
- [ ] Session queue: add 3 prompts → they appear in order
- [ ] Quick launch: opens terminal to correct directory
- [ ] Prompt editor: edit a prompt → changes persist after page refresh
- [ ] Prompt analytics: complete a session → stats update on the prompt card
- [ ] Template prompts: apply "Code Audit" to Chess Coach project → clipboard has the prompt with project context
- [ ] Search prompts: search for "Stripe" → shows all prompts mentioning Stripe
- [ ] Mobile: prompt library is browsable on phone (for reference while at desk)

Show me the prompt library with all your prompts loaded.

SESSION 6: Polish, Packaging & Distribution Prep

Final session. Make Mission Control feel like a real product, package it for easy use, and prep for sharing with other builders.

1. Onboarding / Setup Wizard: First time running Mission Control:

- Step 1: "Where are your projects?" — let user add project directories

    - Auto-detect by scanning for STATUS.md files in common locations
    - Manual add: paste a directory path
    - Scan for git repos on the machine
- Step 2: "Set up notifications" — choose ntfy/Twilio/Pushover, test delivery
- Step 3: "All set! Your dashboard is ready."
- Store config in a local config.json file

2. System Tray / Background Mode (Windows):

- Mission Control should run in the background even when the browser is closed
- Python agent + FastAPI server run as a background process
- System tray icon (Windows) showing status:
    - Green: all agents running normally
    - Yellow: idle agents detected
    - Red: errors detected
    - Click tray icon → opens dashboard in browser
- Use pystray library for the system tray icon
- Runs on startup (optional — add to Windows startup folder)

3. UI Polish: Final pass on every page:

- Consistent spacing, fonts, and colors
- Smooth transitions between pages
- Loading skeletons for every async section
- Empty states that are helpful (not just "No data")
- Keyboard shortcuts:
    - D: go to dashboard
    - P: go to prompts
    - A: go to alerts
    - R: refresh data
    - /: focus search
- Favicon: mission control / radar icon
- Page titles: "Mission Control — Dashboard", "Mission Control — Prompts", etc.

4. Documentation:

- README.md:
  - What Mission Control does (2 sentences + screenshot)
  - Installation: 3 steps max
  - How to add projects
  - How to set up notifications
  - How to launch
  - Architecture overview (mermaid diagram)
  - Privacy: everything runs locally, AI calls go to Claude API only

5. Packaging:

- Create a single installer/launcher:
  - install.bat: checks Python, installs dependencies, creates config
  - start.bat: launches everything (FastAPI + watcher + opens browser)
  - stop.bat: kills all Mission Control processes
- Or: PyInstaller for the Python backend into a single .exe
  - Double-click mission-control.exe → everything starts
  - System tray icon appears
  - Dashboard opens in browser

6. Future Monetization Notes (MONETIZATION.md):

Free (current — personal tool):

- Unlimited project monitoring
- Real-time dashboard
- Basic alerts (ntfy.sh)
- Prompt library

Pro ($9/month — when demand exists):

- AI-powered summaries and recommendations (Claude API costs money)
- Advanced analytics (token burn tracking, productivity metrics, project timelines)
- Twilio SMS alerts
- Team mode (multiple developers monitoring shared projects)
- Webhook integrations (Slack, Discord, email)
- Historical analytics (track productivity trends over months)

Distribution Plan:

- Post on r/ClaudeAI, r/ChatGPT, r/LocalLLaMA — "I built a free dashboard to monitor my AI coding agents"
- Post on Indie Hackers with build story
- Tweet thread showing the dashboard in action with real agent monitoring
- GitHub repo (open source the core, keep Pro features separate)
- Product Hunt launch when polished enough

TESTING:

- [ ] Full new-user flow: run install.bat → setup wizard → dashboard loads with detected projects
- [ ] System tray icon appears and shows correct status color
- [ ] Click tray icon → dashboard opens
- [ ] Close browser → agents still monitored → alerts still fire
- [ ] Reopen browser → dashboard reconnects and shows current state
- [ ] Run 2+ Claude Code sessions simultaneously → both appear on dashboard
- [ ] Complete a session → alert fires → prompt queue suggests next prompt → PROJECTS.md auto-updates
- [ ] Kill Mission Control (stop.bat) → clean shutdown, no orphan processes
- [ ] start.bat after a reboot → everything comes back up correctly
- [ ] No console errors on any page
- [ ] Dashboard runs for 2+ hours without memory leaks or disconnections

Show me the final state of Mission Control monitoring real agent sessions.

TOTAL SESSION MAP

Session	Focus	What Gets Built
1	Status Watcher & API	Python watcher + FastAPI server + log parsing
2	Web Dashboard	Real-time monitoring UI with project overview
3	Alert System	Text notifications via ntfy/Twilio + alert rules
4	AI Integration	Claude-powered summaries + context export for Claude.ai
5	Prompt Manager	Prompt library + queue + quick launch
6	Polish & Packaging	System tray, installer, docs, distribution prep

6 sessions. One per day = usable dashboard in under a week.

This project is unique because it's a META-TOOL — it makes all your other projects more efficient. Build it third (after Finance Brief and Chess Coach are further along) so you have real agents to monitor while developing it.

UPDATED PROJECT LINEUP

#	Project	Sessions	Revenue	Priority
1	AI Finance Brief	8	$9/mo subscriptions	🔴 Active — 6/8 done
2	AI Chess Coach	10	$7/mo subscriptions	🔴 Active — 3/10 done
3	Agent Mission Control	6	Free → Pro tier later	🟡 Start after Brief ships
4	PC Bottleneck Analyzer	6	Free → affiliate + Pro	🟡 Start after Chess ships
5	Trade Journal	7	$15/mo subscriptions	🟣 Sidelined