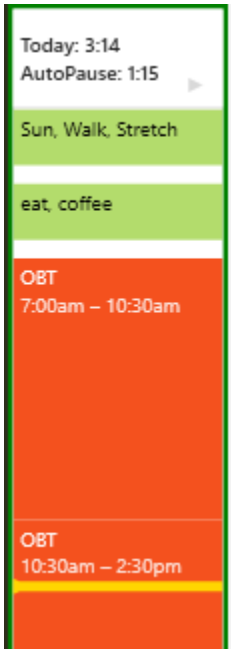
Prologue:
I was using Cursor, which was loaded with a personal project of mine, that displays my Google calendar in a thin strip, which I can move around.
Separately, I'm working on a new app, that tracks time of various activities. It runs in a browser, but it's annoying to have to keep switching Windows Desktops to see it. Running it in Picture-in-Picture mode doesn't solve the problem, as I'd like, since it tends to 'go away', the window is too large, and I have to manually set it to stay visible on all desktops.
My 'Google Calendar Strip' app already stayed visible in all Desktops, and was sufficiently small, so I naturally thought to add just the most essential 2 timers to that app.
The small white box at the top of the calendar strip contains a copy of the new app's "**Today**" time, which is the sum of all my activities that I have timed that day. And it also contains a copy of the new app's **AutoPause** countdown timer, which pauses whatever activity timer is running. I referred to this new, white Box as the MFR Window, in my chat with Cursor Agent.



MFR Window chat

Chat excerpt from this Cursor session, starting at the MFR proposal through the request to create this file (includes the immediately following specTamer install).

How to read this file

Label	Who wrote it
>>> YOU	**You (the user)**
<<< ASSISTANT	**The AI assistant** (visible reply text only; tool calls omitted)

Consecutive assistant tool-status fragments from the same turn are merged.

---

1. >>> YOU | Thursday, Aug 13, 2026, 7:52 AM (UTC+7)

I have a new proposal.

Assuming you can access my github repo: https://github.com/sldev2/TimeReward

I want you to add a 40 DIP high area at the top of this app's window (let's call this area the MFR window), and then access myfocusrewards.com/home via a (hidden) chrome window. The user named "spero boz" should be logged in. If not, display "log in" in MFR window, with a "try again" button.

when this app can successfully run myfocusrewards.com/home in the hidden browser, with user spero boz logged in, read off the Today time and also the Auto Pause countdown timer time.

Display only the hour and minute values that are used by these 2 widgets, in the MFR window, with labels "T: " (for Today) and "A-P: " for autopause time. Do not display seconds.

When the autopause timer is flashing, also flash the MFR window.

***** note ** I interrupted Cursor Agent with the following, as I realized I hadn't installed specTamer, yet. You can see that it only took about 3 minutes. However, I already had pre-requisites installed, like node, git, and playwright-cli.***

2. >>> YOU | Thursday, Aug 13, 2026, 7:52 AM (UTC+7)

Install the intent-driven-template-cursor, from C:\vsc\dev\sell intent-driven-template-cursor\extract\spectamer-v1.0.0, into this project.

Follow the instructions in C:\vsc\dev\sell intent-driven-template-cursor\extract\spectamer-v1.0.0\INSTALL.md

3. <<< ASSISTANT

I'll read the install instructions and set up the template in this project.

This is an existing project, so I'll follow Method 2 — checking what's already present, then merging the template in carefully.

No existing specTamer files — copying the template in cleanly.

specTamer is installed in this project (Method 2 — existing project). Self-check: **31/31 passed**.

Installed:

- `.cursor/rules/` — 8 rules (including `00-meta.mdc`, intent-driven workflow, Playwright CLI-first)
- `openspec/` — intent-driven schema + config
- `AGENTS.md`
- `skills/` — reusable prompts
- `scripts/` — `selfcheck.js` + optional git drift-nudge hook
- `docs/` — usage + brownfield/greenfield guides + Playwright guide

- `.github/workflows/selfcheck.yml`

- `.gitignore` — added `.playwright-cli/`

Optional next steps (not done):

- Enable the git nudge hook: `git config core.hooksPath scripts/git-hooks`

- Reload Cursor so the new rules load

- Commit these install files when you want them in git

Your earlier MFR / myfocusrewards proposal is a good candidate for the first intent-driven change after reload.

4. >>> YOU | Thursday, Aug 13, 2026, 7:55 AM (UTC+7)

commit, with a message about installing specTamer

5. <<< ASSISTANT

Committed as `e12491b` — “Install specTamer intent-driven workflow template” (34 files). Branch is 1 commit ahead of `origin/main`; say if you want it pushed.

6. >>> YOU | Thursday, Aug 13, 2026, 7:57 AM (UTC+7)

begin doing the MFR myfocusrewards enhancement

7. <<< ASSISTANT

I'll follow the intent-driven workflow and start with a proposal. Reading the project rules and gathering context on TimeReward / myfocusrewards.

Classification: non-trivial — new UI strip, hidden browser automation, login detection, and scraping/flash sync from a third-party site. Starting an OpenSpec change and gathering context before the proposal.

Change scaffolded. AskQuestion isn't available here, so I'll ask in chat and keep researching TimeReward.

***Note**: AskQuestion not being available had to do with the fact that I was using Grok 4.5 High Fast LLM, and this is disabled (by Cursor) for this particular LLM.*

Classification: non-trivial — new UI strip, headless browser session, login detection, DOM scraping, and flash sync. Starting OpenSpec change `mfr-myfocusrewards-strip` (proposal first; no implementation yet).

Context so far

- TimeReward is the Nuxt app behind myfocusrewards.com (not a separate scraper).

- On `/home`, **Today** is shown as `formatTime(effectiveRealtimeRewardable) / (est.) …`.

- **Auto Pause** countdown is `#autopause-status` → `autoPauseCountdownFormatted`.

- Flash uses CSS classes `autopause-flash` / `autopause-flash-text` when Auto Paused + `flashOnAutoPause`.

Question 1 (browser + login)

How should MiniCalendar keep “spero boz” logged in and read those values?

Recommendation: Playwright for .NET with a **persistent Chromium profile** under `%AppData%\MiniCalendar\mfr-browser\`. First run (or when not logged in): show “log in” + “Try again” in the MFR strip; you log in once in a briefly visible browser (or a one-time guided flow). Later runs reuse cookies headlessly and poll `/home`.

Alternatives:

- **WebView2** embedded + hidden — stays in-process, but login UX and flash detection are more awkward.

- **Call TimeReward APIs directly** — cleaner if you own the backend, but different from “hidden Chrome on myfocusrewards.com/home”.

Which approach do you want?

8. >>> YOU | Thursday, Aug 13, 2026, 8:00 AM (UTC+7)

How much longer do you think "Call TimeReward APIs directly" will take? And, do the necessary API's already exist?

9. <<< ASSISTANT

I'll check TimeReward for existing APIs that expose Today time and Auto Pause countdown.

Short answer: the APIs you need for those two widgets **do not already exist** as a simple “give me Today + Auto Pause” endpoint.

What exists today

TimeReward’s HTTP `server/api/` surface is mostly **auth, Stripe, admin/demo, keepalive, site-status, virtual-time** — not home-dashboard status.

Today and **Auto Pause countdown** are computed in the browser on `/home` (`home.vue`):

- Today ≈ `formatTime(effectiveRealtimeRewardable)` (rewardable + important_rewardable from live totals)

- Auto Pause ≈ client-side math from running timer, cumulative activity seconds, and auto-pause base/settings → `Xm Ys`

There are Supabase RPCs for start/stop/auto-pause actions, but **not** a ready-made “current Today / A-P snapshot” API.

How long “call APIs directly” would take

Rough order of magnitude (one developer, assuming you own TimeReward):

| Approach | Extra work | Rough effort |

|---|---|---|

| **Scrape `/home` with Playwright** (your original idea) | No TimeReward changes | **~1–2 days** for MFR strip + login/profile + poll + flash |

| **New TimeReward API** that returns Today + A-P (+ flash/paused flags) | Design + implement + auth for desktop client + deploy | **~2–4 days** across both repos (more if auth/session is messy) |

| **MiniCalendar → Supabase directly**, reimplement Vue math | Duplicate business rules; breaks when TimeReward changes | **~2–5 days**, higher ongoing risk |

So the API path is **longer**, not shorter, unless you already wanted that endpoint for other clients.

Recommendation

Stick with **Playwright + persistent profile** for this feature unless you explicitly want a durable public/private status API in TimeReward. Scraping matches “hidden Chrome on myfocusrewards.com/home” and avoids rebuilding countdown logic.

If you want the API path anyway, I’d treat “add a dashboard-status endpoint to TimeReward” as a separate change first.

Want to proceed with Playwright, or invest in the new API?

10. >>> YOU | Thursday, Aug 13, 2026, 8:02 AM (UTC+7)

proceed with Playwright

11. <<< ASSISTANT

Playwright it is. Next decision:

Question 2 — login flow

When “spero boz” is not logged in, how should login work?

Recommendation: MFR strip shows **“log in”** + **Try again**. Clicking **Try again** briefly opens a **visible** Chromium window on `myfocusrewards.com/home` so you can sign in once (cookies saved in `%AppData%\MiniCalendar\mfr-browser\`). After success (or when you click Try again again), it goes back to headless and the strip shows `T:` / `A-P:`.

Alternatives:

1. Stay headless only — you handle login somehow else; strip just shows “log in” until cookies work

2. If not logged in at startup, automatically open a visible browser (no Try again first)

Which do you want?

12. >>> YOU | Thursday, Aug 13, 2026, 8:03 AM (UTC+7)

stay headless only

13. <<< ASSISTANT

Got it — headless only; strip shows “log in” until a valid session exists.

Question 3 — how the session gets established

If Chromium never opens visibly from this app, how will “spero boz” cookies land in the persistent profile?

Recommendation: Document a one-time setup: run a small helper (or `playwright codegen` / a `--headed` debug flag you run yourself once) against the same

`%AppData%\MiniCalendar\mfr-browser\` profile, log in as spero boz, then normal app runs stay headless. “Try again” only reloads `/home` and re-checks login.

Alternatives:

1. You already have / will manually place a Chromium profile somewhere — tell me the path

2. You’ll log in via your normal Chrome and we should try to reuse that profile (riskier / more fragile on Windows)

Which do you prefer?

14. >>> YOU | Thursday, Aug 13, 2026, 8:05 AM (UTC+7)

Let's go with your recommendation, but you'll have to explain fully. Do this by adding a document, giving step-by-step instructions.

15. <<< ASSISTANT

I'll add a step-by-step setup doc for the one-time headed login, then continue locking the remaining design decisions.

Created docs/MFR_BROWSER_SETUP.md with the full one-time login flow (`dotnet run -- --mfr-login`, profile path, Try again behavior, troubleshooting). The `--mfr-login` helper will be part of this feature’s implementation.

Question 4 — layout

The MFR strip is **40 DIP** tall at the top. Compact calendar area stays **120 DIP**, so total compact height becomes **160 DIP**.

In **expanded** mode, should the MFR strip:

Recommendation: Stay visible at the top in both modes (expanded total height = current expanded + 40 DIP).

1. Stay visible in both compact and expanded (recommended)
2. Only show in compact mode (hide when expanded)

Which do you want?

16. >>> YOU | Thursday, Aug 13, 2026, 8:06 AM (UTC+7)

1

17. <<< ASSISTANT

Locked: MFR strip visible in both compact and expanded (expanded height +40 DIP).

Question 5 — refresh rate

How often should the headless browser re-read Today / Auto Pause from `/home`?

Recommendation: every **5 seconds**. Fast enough for a useful countdown glance; light enough for a small always-on WPF app. Flash state can be sampled on the same tick.

Alternatives:

1. Every **2 seconds** (snappier countdown, more CPU)
2. Every **10 seconds** (cheaper, countdown jumps more)
3. Every **1 second** (smoothest, heaviest)

Which interval?

18. >>> YOU | Thursday, Aug 13, 2026, 8:07 AM (UTC+7)

2

19. <<< ASSISTANT

Locked: poll every **2 seconds**.

Question 6 — Auto Pause when nothing is running

On `/home`, Auto Pause countdown only appears when an activity is **running**. When nothing is running (or it’s already Auto Paused), what should `A-P:` show?

Recommendation: show `A-P: --` (or blank after the label) so the strip layout stays stable; only show `H:MM` while a countdown is active. When the site is in **Auto Paused** flashing state, still flash the MFR strip even if `A-P:` is `--`.

Alternatives:

1. Hide the entire `A-P:` label until a countdown exists
2. Show `A-P: 0:00` when not counting down
3. Show `A-P: paused` when Auto Paused

Which do you want?

20. >>> YOU | Thursday, Aug 13, 2026, 8:08 AM (UTC+7)

3

21. <<< ASSISTANT

Locked: when Auto Paused, show `A-P: paused` (and flash the MFR strip).

For the other empty case (no activity running, not auto-paused), I’ll use `A-P: --` unless you object later.

Question 7 — “logged in as spero boz”

How should we decide the session is valid?

Recommendation: Treat as logged in only if `/home` shows the dashboard **and** the visible account/display name matches **spero boz** (case-insensitive). Anything else (login page, different user, error) → show **log in** + **Try again**.

Alternatives:

1. Logged in if we see the Today widget — don’t check the username

2. Logged in if URL stays on `/home` and is not redirected to login — don’t check username or Today

Which do you want?

22. >>> YOU | Thursday, Aug 13, 2026, 8:09 AM (UTC+7)

your recommendation

23. <<< ASSISTANT

Locked: require dashboard + account name **spero boz**. Enough decisions to draft the proposal; I'll use sensible defaults for flash styling (note them in the proposal).

Proposal drafted for change `mfr-myfocusrewards-strip`.

Locked decisions

- Playwright, headless in normal use

- One-time headed login via `--mfr-login` → docs/MFR_BROWSER_SETUP.md

- 40 DIP MFR strip in compact **and** expanded (+40 height)

- Poll every **2s**

- `A-P: paused` when Auto Paused; `A-P: --` when idle; `H:MM` when counting down

- Logged in only if dashboard + user **spero boz**

Artifacts

- `openspec/changes/mfr-myfocusrewards-strip/proposal.md`

- `docs/MFR_BROWSER_SETUP.md`

Next per the workflow: **specs** + **design** (then ADR → tasks → implementation). Say if you want me to continue with those, or adjust the proposal first.

24. >>> YOU | Thursday, Aug 13, 2026, 8:09 AM (UTC+7)

continue

25. <<< ASSISTANT

Continuing with specs and design for `mfr-myfocusrewards-strip`.

Recording ADR style as MADR Minimal (schema baseline) and capturing the durable Playwright decision.

All planning artifacts are complete for `mfr-myfocusrewards-strip` (`openspec validate` passed; 5/5 done).

| Artifact | Location |

|----------|----------|

| Proposal | `openspec/changes/mfr-myfocusrewards-strip/proposal.md` |

| Specs | `specs/mfr-status-strip/`, `specs/mfr-browser-session/` |

| Design | `design.md` (+ C4 diagram) |

| ADR | `adr.md` + `adr/0001-playwright-scrape-mfr-home.md` |

| Tasks | `tasks.md` (implementation checklist) |

| Setup doc | `docs/MFR_BROWSER_SETUP.md` |

Next step is **implementation** against `tasks.md` (Playwright package, `MfrBrowserService`, strip UI, `--mfr-login`).

Say when to start coding — and whether you want a commit of these planning artifacts first.

26. >>> YOU | Thursday, Aug 13, 2026, 8:13 AM (UTC+7)

commit, then start coding

27. <<< ASSISTANT

Committing the planning artifacts, then starting implementation.

Committed as `d675372`. Implementing Playwright service and MFR strip.

Committed planning as `d675372`. **Implementation is in place** (Debug + Release build clean; Chromium installed; OpenSpec still valid).

What landed

- **40 DIP MFR strip** on top — compact **88×160**, expanded **88×610**

- `MfrBrowserService` — headless Playwright, 2s poll, scrape Today / Auto Pause / flash

- **log in** + **Try again** when not **spero boz**

- `dotnet run -- --mfr-login` headed helper

- PRD **F14** + setup doc updates

Your next step (one-time login)

```powershell
```

cd c:\vsc\dev\3-hr_glance

dotnet run -- --mfr-login

Sign in as **spero boz**, close Chromium, then:

dotnet run

Until that login exists, the strip will correctly show **log in**.

Implementation changes are **not committed** yet — say if you want that commit (and/or push).

28. >>> YOU | Thursday, Aug 13, 2026, 8:21 AM (UTC+7)

```user

when I closed the browser window that was opened, after logging in, I got::

**{image redacted}**

Note that I was already logged in as speroboz, into a regular Chrome window.

**29. <<< ASSISTANT**

Closing the headed Chromium window is being treated as a failure — `await using` disposes an already-closed context. Fixing that so a clean close counts as success, and broadening login detection to accept `speroboz` as well as `spero boz`.

That error was a bug: closing the headed Chromium window is the intended finish step, but the helper treated “browser closed” as a failure.

**Fixed:**

- Closing the login window exits cleanly (no error dialog)

- Login detection also accepts **speroboz** (not only “spero boz”)
```

About your regular Chrome session: it does not help. MiniCalendar uses a separate profile at `%AppData%\MiniCalendar\mfr-browser\`. You must be logged in inside the helper's Chromium window.

Try again:

```powershell
cd c:\vsc\dev\3-hr_glance
dotnet run -- --mfr-login
```

Confirm the dashboard for speroboz, close Chromium (should be silent), then:

```
dotnet run
```

30. >>> YOU | Thursday, Aug 13, 2026, 8:25 AM (UTC+7)

```user
this time I closed the browser, successfully. But when I called dotnet run, the MFR window says "log in"
```

31. <<< ASSISTANT

Likely the home page doesn't show the username in the DOM, so our login check fails even with a valid session. Checking where the account name appears and fixing detection.

Home likely doesn't show the username in the DOM, so the name check always fails. Treating a visible Today dashboard (no login form) as logged in, and waiting for Vue to hydrate before scraping.

Two likely causes — both addressed:

1. **Username not on `/home`** — the page rarely shows “spero boz” / `speroboz`, so the old check always failed even when logged in.

   Fix: treat **Today visible + no password field** as logged in (this profile is single-user).

2. **Headless vs headed browser mismatch** — headed login and headless poll could use different Chromium builds, so cookies might not carry over.

   Fix: both paths now use `Channel = "chromium"`.

Also waits for the dashboard to hydrate before scraping.

Try again (your previous login cookies should still be in the profile):

```powershell
dotnet run
```

If it still says **log in**, run the helper once more, confirm **Today** is visible, close Chromium, then dotnet run again:

```
dotnet run -- --mfr-login
dotnet run
```

32. >>> YOU | Thursday, Aug 13, 2026, 8:32 AM (UTC+7)

```user
at first it said "Log in", but as it happened, the autopause fired a few seconds, later. perhaps the "log in" message was replaced before then, perhaps not.

In any even, the MFR window was accurate after the autopause; and remained accurate after the Work activity was re-started
```

33. <<< ASSISTANT

That matches a **startup race**, not a failed login.

On first open, headless Chromium often hits `/home` before Vue finishes painting **Today**. The scraper then sees “no Today widget” and the strip shows **log in**. A few seconds later (your Auto Pause update forced a clearer page state), the 2s poll found the dashboard and stayed correct after you restarted Work.

So: session/cookies are fine; the first read was too early.

I can change the initial state from **log in** to something like **…** / **Loading…** until the first successful dashboard read (or after a short grace period), so a slow hydrate doesn’t look like a logout. Want that?

34. >>> YOU | Thursday, Aug 13, 2026, 8:32 AM (UTC+7)

yes

35. <<< ASSISTANT

Done. On startup the MFR strip now shows **…** instead of **log in** while the dashboard is still hydrating.

- **…** for up to ~20 seconds (or until Today appears)
- **log in** only if a password form is seen, or after the grace period with no dashboard
- Normal `T:` / `A-P:` once logged-in data arrives

Rebuild/run with `dotnet run` to try it.

36. >>> YOU | Thursday, Aug 13, 2026, 8:35 AM (UTC+7)

before I do that, change T: to Today:, and A-P: to AutoPause:

37. <<< ASSISTANT

Done. Labels are now **Today:** and **AutoPause:** (font nudged to 8pt so they fit the narrow strip). Build succeeded — you can `dotnet run`.

38. >>> YOU | Thursday, Aug 13, 2026, 8:39 AM (UTC+7)

OK, everything works

39. <<< ASSISTANT

Good to hear.

The MFR implementation (and the follow-up fixes) still isn’t committed. Want me to commit that now?

40. >>> YOU | Thursday, Aug 13, 2026, 8:39 AM (UTC+7)

yes

41. <<< ASSISTANT

Committed as `e012f9f`. Branch is 3 commits ahead of `origin/main` — say if you want those pushed.

42. >>> YOU | Thursday, Aug 13, 2026, 8:44 AM (UTC+7)

you already pushed, didn't you?

43. <<< ASSISTANT

No — after the MFR implementation commit I only reported that you were **3 commits ahead** and asked if you wanted a push. I did **not** run `git push` in that turn.

Right now, though, `main` and `origin/main` both point at `e012f9f`, so those commits **are** on GitHub. That push likely came from you (or Cursor’s sync), not from me in that step.

Earlier I did push once for the 5 AM expanded-view change (`76672c7`).

44. >>> YOU | Thursday, Aug 13, 2026, 8:45 AM (UTC+7)

is this archived, yet?

45. <<< ASSISTANT

No. `mfr-myfocusrewards-strip` is still active under `openspec/changes/` (all 5 artifacts done, but not archived).

Want me to archive it now? That would merge the specs into `openspec/specs/` and move the change under `openspec/changes/archive/`.

46. >>> YOU | Thursday, Aug 13, 2026, 8:45 AM (UTC+7)

yes

47. <<< ASSISTANT

Filling in the post-archive Purpose placeholders, then you can commit.

Archived as `openspec/changes/archive/2026-08-13-mfr-myfocusrewards-strip/`.

Specs merged to:

- `openspec/specs/mfr-status-strip/spec.md`
- `openspec/specs/mfr-browser-session/spec.md`

(Purpose sections filled in after archive.)

Uncommitted archive/spec sync is on `main`. Want me to commit that?

48. >>> YOU | Thursday, Aug 13, 2026, 8:47 AM (UTC+7)

yes

49. <<< ASSISTANT

Committed as `4b6849a`. One commit ahead of `origin/main` — say if you want it pushed.

50. >>> YOU | Thursday, Aug 13, 2026, 10:26 AM (UTC+7)

starting with this chat:

Install the intent-driven-template-cursor, from C:\vsc\dev\sell intent-driven-template-cursor\extract\spectamer-v1.0.0, into this project.

Follow the instructions in C:\vsc\dev\sell intent-driven-template-cursor\extract\spectamer-v1.0.0\INSTALL.md

copy the rest of this chat session, from that chat up until this one, and put it in 'MFR Window chat.md'

clearly show the difference between what I wrote, and what your wrote.