

ProdKit

Async standup digest for small distributed game teams

Design and build process document · Farnoud Fathi

What this document is

A record of how a production tool was designed and built — the decisions taken, the reasoning behind each, the alternatives rejected, and the problems found along the way. It exists because the reasoning is the part worth showing. Anyone can install a standup bot; the useful question is why this one works differently, and what was learned building it.

Why this was built

Across several teams — remote, partly remote, and co-located — the same failure kept recurring, and after enough of them it stopped looking like bad luck and started looking like a pattern worth solving.

The board did not reflect reality. People weren't opening the tracker, so statuses drifted, and every report built on top of them inherited the drift. Standups reported what people remembered rather than what was true. The producer's picture of the project was always a few days stale, and the gap showed up as a surprise late in a sprint rather than as a warning early in one.

The usual fixes make it worse. Asking people to log hours doesn't work — they forget, they backfill from memory at the end of the week, and it reads as surveillance. Chasing people individually doesn't scale, and it turns a producer into an interruption. Adding a second tool for status adds a second thing nobody opens.

What the pattern actually says is that the tracker loses to the work. Anyone mid-task will choose the task over the admin, every time, and they are right to. So the constraint was set deliberately:

Make the board more accurate while asking the team to do less, not more.

The problem being solved

On a small distributed team the producer has two bad options: ask people to log hours and update tickets — they won't, reliably, and it feels like surveillance — or fly blind on where work actually stands.

Artists in particular rarely open the tracker. So the board goes stale, and every tool built on top of it inherits that staleness. Any solution has to make the board more accurate while asking the team to do less.

The core idea

Jira already records every status change with a timestamp, for free, whether or not anyone logs hours. That record is enough to reconstruct how long work actually took. And if answering the standup writes back to the board, the team does strictly less work than before while the board becomes accurate for the first time.

Those two ideas — **derive from the changelog, write back from the answer** — are what separate this from an off-the-shelf standup bot.

Design decisions

Each of these was argued through before any code was written. The build order was deliberately design-first: agree what the tool does, stress-test it, then build.

Scope and approach

D1 Merge the standup bot and the digest bot into one tool

CHOSE One product: collect answers, then post a digest combining them with board state.
WHY Async standup collection alone already exists (Geekbot, Standuply). The differentiator is the Jira layer — human input plus system state in one post. Split across two tools, neither half is interesting.

D2 Write-back to Jira on button tap

CHOSE	Button taps transition tickets, add labels, and post comments.
WHY	Without it the tool is a nicer standup form — the person still has to open Jira afterwards, so it adds a step rather than removing one. With it, the standup IS the board update.
COST	Requires handling failed writes visibly, and an undo path when someone mis-taps.

D3 Buttons for status, one optional text field for blockers

CHOSE	Three buttons per ticket; a text box opens only when someone reports being blocked.
WHY	Target under 30 seconds with no typing. The moment it costs more, adoption dies — and the people most likely to abandon it are exactly the ones whose updates matter most.

D4 Code for the standup bot, no-code for the future playtest pipeline

CHOSE	Python plus discord.py here; n8n or similar for form-to-ticket routing later.
WHY	The standup tool needs business-hours arithmetic, scheduled jobs and changelog parsing — no-code chokes on all three. Form routing is exactly what no-code is for. Choosing per tool rather than per portfolio is the more honest demonstration.

The bot/producer split

The single most useful rule in the design, and it arrived late — after an argument about which tickets deserve to be asked about:

The bot asks only what cannot be read off the board. The digest reports what can.

Signal	On the board?	Channel
Is this active ticket actually finished?	No	Ask
Did the handoff actually happen?	No	Ask
Can you start this today?	No	Ask
Ticket untouched for two working days	Yes	Flag to producer
Overdue or due soon	Yes	Flag to producer
Blocker still open	Yes	Notice, no tap

This resolved a problem that had been circling for several rounds: how to rank tickets when only a few can be asked about. Once staleness moved to the digest, the ranking question mostly dissolved — every active ticket gets the same neutral question, and the producer handles staleness human-to-human. Nobody gets nagged by a bot about something their producer could already see.

Handoff verification

The feature nothing off-the-shelf does, and it came from a real failure on a real team rather than from a feature list. A blocking ticket being marked Done does not mean the person waiting received anything. The retopo is finished, the ticket is closed, and the file never reached the rigger. Both tickets look healthy. Work is stopped. Nobody finds out until someone asks.

Blocker state	What the bot does
Still open	Reports it as a notice. No question — the person genuinely cannot act, so asking would be noise.
Marked Done	Asks: do you have what you need to start? Three buttons — got it and starting, got it but not today, still haven't received it.
No links	Nothing. Silence rather than an empty section.

D5 Handoff asks once, failure repeats daily with escalation

CHOSE	"Got it" settles the question permanently. "Haven't received it" re-asks each morning, and the digest counts the days.
WHY	Only the unresolved case should stay loud. Day one is a note; day three is 27 hours of a rigger idle on work everyone believes is finished, and it should read that way.

D6 "Got it" does not write to Jira

CHOSE	Only "got it, starting today" transitions the ticket.
WHY	Receiving a file is not starting work. Transitioning on "got it" would fill In Progress with work nobody began, and the cycle-time tool would count hours against untouched tickets. The button that moves a ticket now says exactly what it means.

Detecting idle people without asking

The most valuable signal in the tool costs zero taps. If everything someone has active is done or blocked, and everything in their To Do is blocked, they have nothing to work on. That is derivable from answers already given.

Deriving it is also more reliable than asking. Nobody volunteers "I have nothing to do." It becomes the top line of the digest — a producer finding out at 13:00 instead of at Friday's check-in.

Jira structure

D7 Company-managed project, not team-managed

CHOSE	Company-managed Scrum board.
WHY	Components don't exist in team-managed projects — the workarounds are custom fields, labels, or a paid plugin. Components are needed for bug routing later, and they can carry a default assignee, which gives auto-routing for free. Company-managed also exposes configurable workflows, which matters when writing transitions through the API.
COST	Slower setup, self-administered.

D8 Blocked as a label plus issue links, not a column

CHOSE	Tickets stay in the status reflecting their real state; blocked-ness sits on top.
WHY	A Blocked column destroys cycle-time data — a ticket bouncing In Progress → Blocked → In Progress leaves the active state, so duration depends on whether anyone remembered to move it. A column also loses information: blocked out of To Do is a scheduling problem, blocked out of In Progress is someone idle right now. Issue links go further and name which ticket is blocking which.
COST	Less visible at a glance than a column. Acceptable, because the digest surfaces it daily.

D9 Estimate in hours, not story points

CHOSE	Original estimate in hours, optional per ticket.
WHY	The engine measures actual business hours from the changelog — hours estimated against hours measured is like for like. Points would need a conversion factor, and the standard guidance is that once points are translated to hours they aren't points any more. Points also need several sprints of velocity to calibrate; a five-day run never does. And most of the team are artists, whose work is time-predictable in a way engineering discovery is not.
COST	Hour estimates can be misused to judge individuals. Mitigated by design: the digest never shows per-person throughput, and overruns are reported per ticket.

Review step

Raised late, after write-back was working: should the assignee be able to move their own ticket to Done? The answer shapes whether the In Review column means anything.

Standard practice is that the person doing the work signals completion and someone else confirms it — the assignee moves to In Review, a reviewer or lead moves to Done. The distinction is "I've finished my part" versus "this meets the bar." If assignees can self-close, the review column is decorative.

D10 Review is an optional feature, off or on at setup

- CHOSE** When enabled: the button reads “Ready for review” and transitions to In Review; the ticket is reassigned to the component lead read from Jira; the reviewer gets approve / send back / not yet buttons in their own standup.
- WHY** Some teams have no review step, and forcing one would make the tool wrong for them. Reading the reviewer from Jira’s component lead rather than duplicating it in config avoids two sources of truth going out of sync.
- COST** Reassignment moves the ticket off the original assignee, so attribution has to be recovered from the changelog — the same technique already used for cycle time.

Demo and testing approach

D11 Threads in one channel rather than direct messages

- CHOSE** The bot opens one visible thread per person inside a single channel. DM mode exists and is one config value away.
- WHY** The entire flow is capturable in a single frame for portfolio video, and nobody’s private messages are exposed. The code path is identical either way — only the delivery target changes.

D12 Build solo, bring the team in only for the final run

- CHOSE** Verify the full loop alone against a board where every ticket is self-assigned, then run a week with the team.
- WHY** Several people’s goodwill is a limited resource — realistically one shot at asking for a week of their mornings. Debugging in front of them spends it for nothing.

D13 Synthetic demo data, labelled as such

- CHOSE** Fictional project with generated history, fixed random seed for reproducibility.
- WHY** Standard practice for tool demos, and it lets the portfolio piece run without credentials. The rule: the tool is real, the data is labelled. Metrics derived from invented data never appear on a résumé as achievements.

Architecture

Six layers. The separation is load-bearing: the parts that know about Discord, about Jira, and about standup logic are deliberately kept apart so each can be replaced without touching the others.

File	Responsibility
config.yaml	Everything team-specific — column names, channel IDs, roster, schedule, caps. No project details live in code, which is what makes it deployable elsewhere.
config.py	Loads and validates. Fails at startup with a clear message rather than three hours later when the digest posts into nothing.
models.py	Plain data shapes — Issue, BlockerRef, Person, WorkCalendar. Nothing downstream touches raw API JSON.
business_time.py	Working-hours arithmetic against each person’s own calendar and timezone.
jira_client.py	The only file that knows Jira’s API shape. Reads the board, resolves blockers, reads changelogs, writes transitions and comments.
questions.py	Pure logic. Turns one person’s tickets into their message set and the flags their board state raises. No network, so it is testable at a terminal.
writeback.py	Maps button actions to board operations, and defines which actions deliberately do nothing.
digest.py	Assembles the producer digest, ordered by urgency.
scheduler.py	Per-person local times for prompt, nudge, cutoff and end of day.
bot.py	Discord only. Threads, buttons, modals. Takes Question objects and never touches Jira directly.

Why business hours, not wall clock

A ticket moved to In Progress on Friday at 16:30 and untouched until Monday at 10:15 shows 65.75 hours elapsed. Real working time is 2.75 hours.

Report the first number and every ticket that touches a weekend looks abandoned. The digest fills with false alarms and stops being read within a week. So every duration is measured against the assignee's own calendar — and on a distributed team the same interval differs per person. That Friday-to-Monday span is 2.75 working hours in Vancouver and 4.25 in Toronto, because Friday 16:30 Pacific is already past end of day Eastern.

The implementation walks the interval one local day at a time rather than doing modular arithmetic, which handles daylight saving correctly for free — each day's 09:00 and 18:00 are constructed by the timezone library instead of assuming every day is 24 hours long.

Why buttons survive restarts

Discord stores the button's identifier but not the code that handles it. A normal button lives in the bot's memory, so after a restart every previously sent button reports "This interaction failed."

The fix is to make handlers stateless: everything needed to process a tap is encoded in the identifier itself.

```
su:2026-08-20:got_starting:SR71-17
```

Date, action, ticket. The bot needs no memory of what it sent and rebuilds the handler from the string. A worthwhile clarification found during testing: the widely-quoted 15-minute limit applies to the response window after a click, not to the button's lifetime. Buttons stay live indefinitely — but the bot process must be running when someone taps, which is a hosting requirement, not a code one.

Problems found and fixed

Kept deliberately, because the failures are more informative than the successes.

During build

Problem	Cause	Fix
Half the demo board flagged as stalled at 3800% of estimate	Every synthetic ticket aged uniformly over eight weeks, so tickets "in progress" had been so since January	Age tracks outcome — completed work can be old, active work is days old
Sprint warning: seven tickets missing an estimate	Board's estimation statistic was Story Points while the tool reads Original Estimate — two parallel systems, one empty	Switched the board statistic to Original Time Estimate so Jira's charts and the tool agree
Every timezone lookup failed on Windows	Python reads the OS timezone database; Windows ships none	Added the tzdata package; added a timezone check to the setup validator
Handoff messages read "marked done by someone"	Jira's issue-link payload carries only a stub of the linked ticket — no assignee	One extra query resolving every blocker key at once, rather than one lookup each
Running the post command twice created duplicate threads	Manual and scheduled paths used different guards	Both paths now write and check the same per-day marker
Buttons dead after closing the terminal	Discord needs an acknowledgement within three seconds; no process was running	Confirmed as a hosting requirement rather than a bug — persistent buttons survive restarts, not downtime
Interesting tickets cut by the question cap	Ranking by due date only; undated tickets tied and fell back to key order	Reframed: questions are capped, flags are not. Staleness moved to the digest

During the live run

Eight further problems surfaced only once real people were using it on their own schedules — which is the argument for running it with a team rather than declaring it finished at the end of the build.

Problem	Cause	Fix
Correcting a mis-tap reset the staleness clock	Any transition counted as movement, so an undo looked like activity	Round trips inside a 30-minute window collapse; repeated corrections surface in the digest instead
Digest posted for the wrong day when it ran at 17:00	Dates derived in UTC, which had already rolled over	All dates derive from the project timezone
The oldest review ticket was dropped by the cap	Reviews ranked by due date, which reviews often don't have	Ranked by how long they have been waiting
Nudge and end-of-day messages reached nobody	Both posted into the standup thread, which people had stopped opening	Both DM first and link the thread
Nudges went to people who had already answered	The check ran when the job was scheduled, not when the message was sent	Moved to the point of send
Notes typed during standup vanished	Notes and the finish button shared one storage key	Notes write to a separate key
Reviewers were flagged as days overdue on other people's work	Review tickets counted toward the reviewer's own deadline flags	Review tickets excluded from personal deadline flags; a review queue still counts as work
Slash commands reported success and changed nothing	The scheduler held a config reference captured at construction, so it never saw an update	The scheduler reads config through the bot rather than holding its own copy

Reversals

Two positions changed during the process, both worth recording because the reasoning moved rather than the preference:

Ranking questions by urgency. Abandoned once the ask/flag split existed. Urgency is visible on the board, so it belongs in the digest; the limited question slots should go to things only a person can answer.

No undo on answers. Originally listed as an accepted limitation. Reversed on the argument that a known failure cheap to prevent shouldn't be left to a README — someone will mis-tap on day one, and the recovery cost is worse than one extra button.

Deliberate limitations

Stated rather than hidden. Knowing what a tool cannot do is part of knowing what it does.

- A ticket forgotten in In Progress accrues active hours whether or not anyone is touching it. The tool cannot distinguish “being worked on” from “left in the wrong column” — which is exactly why stall detection sits alongside the raw number rather than replacing it.
- The blocked-lookahead depends on issue links being maintained. Nobody maintains them automatically, so the producer does. When no links exist the digest says so explicitly rather than showing an empty section that reads like good news.
- Undoing a write-back needs the status the ticket came from, held in memory. If the process restarts between the tap and the undo, the bot says so and asks the person to fix it manually rather than guessing — a wrong guess would corrupt the changelog the cycle-time tool depends on.
- Setup friction is real: a Jira API token, a Discord bot token, and an identity mapping. Roughly twenty minutes of unglamorous work, and where most self-hosted tools lose people.

Build status

Step	State	Note
1 · Connection check	Done	Verifies both tokens; dumps every Discord and Jira ID needed for config
2 · Config and validation	Done	Ordering, duplicate IDs, timezones and column mapping all checked at startup

Step	State	Note
3 · Jira read layer	Done	Board fetch, blocker resolution, changelog reading, pagination
4 · Question builder	Done	Pure logic, terminal-testable; handoff branching verified both ways
5 · Discord prompts	Done	Threads, persistent buttons, modals, change-answer, duplicate guard
6 · Write-back	Done	Transitions, labels, comments, undo, visible failure reporting
7 · Session store	Done	Answers persist on a mounted volume; a restart doesn't lose the morning
8 · Review feature	Done	Feature flag, reviewer read from component lead, approve / send back buttons
9 · Digest assembler	Done	Ordered by urgency: idle people, then handoff failures, then blockers
10 · Scheduler	Done	Per-person local times: prompt, nudge, cutoff, end of day, close
11 · Setup validator	Done	Checks config against the live board and server before a run
12 · Hosting	Done	Containerised, persistent volume, runtime settings survive redeloys

Feature-complete and deployed. Ran unattended for a week against a live Jira board with a six-person team. That run surfaced eleven bugs across scheduling, storage and state handling, each documented above with its cause and fixed.

Where this goes next

A second tool — cycle-time analysis — reconstructs how long work actually takes from the same changelog data, without anyone logging hours. The digest assembler was built to accept board-state flags it does not yet produce, so that tool can extend it rather than force a rewrite.

The two are worth more together than apart, and the reason is the point of the whole project: cycle-time measurement is only as good as the accuracy of status transitions, and the standup bot is what makes those transitions real.

Project names, tickets and team members shown in development screenshots are from a rehearsal board and are labelled as such. No metric in this document is derived from invented data.