TASKCOMPASS

Task Tracking App for Developers

Run sprints, backlogs, and PR handoffs with a task tracking app developers will actually open every morning.

Adrian Mercer, Senior Editor · 11.05.2026

TL;DR A task tracking app for developers lives or dies by three things: keyboard speed, tight Git integration, and a board that engineers will open without being asked. Linear has eaten the share that Jira used to own at small-to-midsize startups; Jira still dominates regulated and large-org settings; Height and Shortcut occupy a niche between them. Linear Free covers 250 issues and 2 teams; Basic is $10/user/mo and Business $16/user/mo. Pick for the engineering culture you have, not the methodology a consultant sold you.

Agile Development Workflows

Most engineering teams over-implement agile and under-implement the parts that move cycle time. The right workflow is the one your engineers will follow without a Scrum Master enforcing it.

Agile in 2026 looks very different from agile in 2016. The teams shipping fastest run lighter rituals, tighter Git integration, and far less estimation theatre.

Pricing and feature data verified against vendor pages on May 14, 2026.

Sprint-based versus continuous-delivery flows

Sprint cadence still earns its keep when you have stakeholders who need predictable release windows. Continuous delivery wins for teams whose customers consume changes immediately. Linear's "Cycles" abstraction supports both without forcing a re-setup; Jira leans sprint-first; Shortcut and Height blur the line. A GitHub-linked task app like Linear is the path of least resistance if your team already lives in pull requests.

Mapping issues to commits and pull requests

Every modern engineering issue tracker now auto-links commits and PRs by issue ID in the branch name. Linear's "magic words" close issues from PR descriptions; Jira does the same with smart commits. The win is not the link — it is that engineers stop manually updating ticket state.

Avoiding agile theatre on small dev teams

Below 8 engineers, a daily standup, sprint planning, retro, and grooming session in the same week often consumes 15% of engineering capacity. Cut to one short weekly priorities meeting plus an async update; reinstate ceremonies only when team size or release cadence demands them.

- Continuous flow until 8 engineers, then evaluate sprint cycles
- Use issue-ID branch naming so PRs auto-close issues
- Drop ceremonies that consume more than 10% of engineering time

Run the lightest agile process your release cadence allows; let Git integration do the bookkeeping.

Sprint Planning and Backlogs

Sprint planning fails when the backlog is a graveyard of stale tickets. Half the planning improvement comes from grooming, not from better estimation methods.

Most engineering teams carry 3-5x more open issues than they will ever ship. The bigger that pile gets, the harder planning becomes. The fix is structural, not behavioural.

Story points, hours, or no estimates at all

Story points work when the team is stable and the velocity number is used for forecasting, not performance review. Hours work for client billing. "No estimates" with strict WIP limits and a target throughput works for continuous-delivery teams. Linear nudges teams toward T-shirt sizes (S/M/L/XL); Jira and Shortcut expose the full range. The dev backlog tool does not pick the method — the team does.

Grooming a backlog without endless meetings

Async grooming beats synchronous grooming. Engineers tag stale issues with "needs-decision" or "stale" weekly; a 20-minute Friday review clears the queue. Tickets older than 90 days with no recent activity should auto-close — Linear has this built in.

Tracking carry-over and unplanned work

Two metrics matter: percentage of the sprint carried over (target under 20%) and percentage of work that came in unplanned (target under 25%). If unplanned is over 40%, the team is operationally reactive — fix that before tweaking estimates.

1. Pick one estimation method and stick with it for at least three months
2. Auto-close issues older than 90 days with no activity
3. Watch carry-over and unplanned work as your two real planning signals

A clean backlog and consistent estimation method matter more than which estimation method you pick.

Collaboration for Engineering Teams

Engineering collaboration breaks down at the seams: between product and engineering, between time zones, and between the issue tracker and the code review tool. The right team workflow app reduces those seams.

Most "collaboration problems" are actually handoff problems. Where an artifact moves between people or systems, friction accumulates. Map the handoffs and the fix becomes obvious.

Cross-functional tickets with product and design

Product writes the brief, design produces the mockup, engineering implements, QA verifies. If those four states live in four different tools, the ticket loses context at every transition. Linear,

Jira, and ClickUp all support custom workflow states; pick one tool where the whole loop lives, even if non-engineers grumble.

Code review tasks alongside feature work

Pull request review is engineering work and should appear on the engineer's task list. Linear's PR view and Jira's GitHub-for-Jira app surface review queues alongside assigned issues. Engineers who treat reviews as bonus work let them slip; those whose tools surface reviews next to issues do not.

Async handoffs that survive timezone gaps

Distributed teams need three things: clear ticket descriptions, recorded design walkthroughs, and a "blocked on" field that names the blocker. The cost of an async handoff is one extra paragraph; the cost of a failed handoff is 24 hours of nothing happening. Time zone handoffs are where remote-team productivity is won or lost.

- Keep cross-functional tickets in one tool, even if non-engineers prefer another
- Surface PR review queues next to assigned issues
- Spend the extra paragraph on async handoffs to avoid 24-hour stalls

> *Reduce handoff seams; one tool covering product, design, engineering, and QA beats best-of-breed in four.*

Integrations and APIs

Integration depth, not feature checklists, separates the tools developers will tolerate from the ones they will adopt. The API quality is also the quality of the third-party ecosystem you inherit.

Every modern issue tracker advertises GitHub, GitLab, Slack, and Figma integrations. The differences show up in latency, in two-way sync depth, and in what you can build on top through the API.

GitHub, GitLab, and Bitbucket integrations

Linear's GitHub integration is the most often praised: branch naming, PR linking, and status sync are bidirectional and fast. Jira's GitHub-for-Jira app covers the same ground but is usually heavier to administer. GitLab integrations are first-class on Jira and acceptable on Linear; Bitbucket integration is most natural inside the Atlassian stack.

CI/CD status pinned to the task board

The valuable signal is "is this issue's branch passing CI?" — surfaced inline on the issue page. Linear shows GitHub Actions status next to PRs; Jira surfaces Bitbucket Pipelines and (via app) GitHub Actions. Shortcut and Height both show CI status; the polish gap with Linear is small.

How good is each tool's public API?

Linear's GraphQL API is the best-documented and most consistently versioned of the four. Jira's REST API is comprehensive but its versioning is inconsistent across endpoints. Shortcut's REST API is clean but smaller in surface area. Height has good API coverage and recently added webhooks for most state changes.

- Branch-naming auto-link works in all four; Linear has the lowest latency
- CI status next to PRs is universal in 2026; differences are cosmetic
- Linear's GraphQL API is the most pleasant to build against

> *Integration latency and API quality decide which tool engineers will actually live in day-to-day.*

Developer Productivity Features

Developer productivity inside a task tracker comes from three things: a fast keyboard interface, decent markdown, and command-palette navigation that does not make you reach for the mouse.

The features developers actually use every day are dull and load-bearing: code blocks that render with syntax highlighting, a command palette that opens any view, and keyboard shortcuts for state changes. The tool that gets these right gets opened.

Markdown, code blocks, and syntax highlighting

Linear and Height ship full markdown with syntax-highlighted code blocks out of the box. Jira's markdown is partial and frequently complained about. Shortcut handles markdown well. For an engineering issue tracker, this matters more than it sounds — issue descriptions full of stack traces are unreadable without proper code formatting.

Keyboard-first navigation and command palettes

Linear's command palette (Cmd+K) is the gold standard; nearly every action has a shortcut. Height ships a similar palette. Jira added a command palette in 2024 and it is now competent. ClickUp's keyboard story is weaker — the tool was designed mouse-first.

Linear, Jira, Height, and Shortcut compared

Tool	Free tier	Cheapest paid	API	Keyboard-first
Linear	Free up to 250 issues, 2 teams	Basic $10/user/mo	GraphQL, well-documented	Yes (best in class)
Jira	Free up to 10 users	Standard ~$7-$8/user/mo band	REST, comprehensive	Yes (added 2024)
Height	Free tier available	Paid tier	REST + webhooks	Yes
Shortcut	Free up to 10 users	Paid tier	REST, clean	Partial

Linear Business at $16 per user per month unlocks unlimited teams; below that, the 5-team cap on Basic is the constraint that pushes most growing engineering orgs to upgrade.

Editor's note: For a 12-person engineering-team scenario, Linear's biggest advantage over Jira is not a longer feature list. It is the command palette, keyboard-first state changes, and lower setup overhead. Jira remains stronger when governance, custom workflows, and Atlassian reporting matter more than triage speed.
— Adrian

Keyboard speed, decent markdown, and a real command palette decide whether engineers open the tool unprompted.

FAQ

Is Linear or Jira better for a small engineering team?
For teams under 50 engineers without compliance requirements, Linear is the modern default — faster keyboard interface, cleaner GitHub integration, and a free tier that covers 250 issues and 2 teams. Jira is the better pick for regulated industries, large orgs needing deep customisation, or teams already on the Atlassian stack with Confluence and Bitbucket. Linear Basic is $10/user/mo; Jira Standard sits in the $7-$8 band.

Do developers actually use story points?
Mixed. Story points work when a stable team uses velocity for forecasting, not performance review. Many high-velocity teams use no estimates at all and rely on strict WIP limits plus throughput metrics. T-shirt sizes (S/M/L/XL) are the most common compromise in 2026 — fast to assign, useful for forecasting, and resistant to the gaming that hour-based estimates invite.

How does PR-to-issue linking work in Linear and Jira?
Linear uses branch naming (e.g. ENG-123-feature-name) plus magic words in PR descriptions like "Closes ENG-123" to auto-update issue state when PRs merge. Jira uses smart commits with the same idea. Both update bidirectionally — issue state moves when the PR moves. Linear is faster end-to-end; Jira via the GitHub-for-Jira app lags by 30-60 seconds in real-world testing.

Should engineering teams use one tool for product, design, and engineering work?
Yes, in most cases. The cost of cross-tool handoffs (lost context, manual status updates, broken links) is higher than the cost of asking non-engineers to use an engineering-flavoured tool. Linear, Jira, and ClickUp all support custom workflow states that can model design and product work alongside engineering. The tool with the deepest GitHub integration is usually the right hub.

What is the cheapest paid plan that scales past 5 teams?
Linear Business at $16 per user per month unlocks unlimited teams; Linear Basic at $10 caps at 5 teams. Jira Standard sits in the $7-$8 band and supports more teams from the start but lacks Linear is keyboard-speed. ClickUp Unlimited at $7 per user per month also has no team cap. The 5-team ceiling on Linear Basic is the most common upgrade trigger for growing engineering orgs.

Full article: https://tasktrackingapp.net/task-tracking-app-for-developers