

# **GEO Analysis Report**

Generative Engine Optimization

**Linear**

January 18, 2026

Prepared by LLMEposure

## Contents

---

<b>I. About This Report</b>	<b>2</b>
What is GEO? . . . . .	2
What We Did . . . . .	2
How We Calculate Scores . . . . .	2
The Honest Truth . . . . .	3
What's in This Report . . . . .	3
<b>II. Scorecard</b>	<b>4</b>
Linear Performance . . . . .	4
Competitive Comparison . . . . .	4
<b>III. Analysis</b>	<b>5</b>
GEO Analysis Deep Research Report: Linear . . . . .	5
Executive Summary . . . . .	5
How You Compare . . . . .	6
Why You're Not #1 . . . . .	7
Query Analysis (When You Show Up vs. When You Don't) . . . . .	9
The Diagnosis: GEO vs. SEO . . . . .	11
Specific Recommendations . . . . .	12
1. Create a “Linear vs. [Competitor]” Content Hub . . . . .	12
2. Add “How Linear Handles X” Guides (Deep Dives on Key Topics) . . . . .	13
3. Publish a “State of Execution” Report (Original Research & Thought Leadership) .	14
4. Refresh Messaging to Address “Why not Linear” upfront . . . . .	15
5. Leverage Community Content and Schema for Better AI Indexing . . . . .	16
6. Monitor Future AI Responses and Iterate . . . . .	16
Sample Conversations . . . . .	17

## I. About This Report

---

### What is GEO?

Generative Engine Optimization—GEO—is the practice of influencing how AI assistants like ChatGPT, Claude, and Perplexity talk about your company. When a potential customer asks an AI “What are the best options for X?”, will your company be mentioned? Will you be recommended first, or buried at the bottom, or left out entirely?

This is a new frontier, and we’ll be honest with you: **GEO is still a dark art.** Nobody—not even the labs building these models—fully understands why they say what they say. We can measure, probe, and experiment, but anyone claiming to have cracked the code is overselling.

That said, we *can* learn a lot by systematically testing how models respond to real customer questions. That’s what this report does.

### What We Did

We simulated **200 conversations** that your potential customers might realistically have with an AI assistant.

For each conversation, we:

1. Took on the persona of someone in your target market
2. Asked a question they might genuinely ask when looking for solutions
3. Requested the AI recommend companies that could help
4. Tracked whether **Linear** was mentioned, and at what position
5. Asked follow-up questions to understand *why* you were or weren’t recommended

We ran every conversation **twice**: once with the AI’s web search disabled (testing what the model “knows” from training), and once with web search enabled (testing your current SEO and web presence). The gap between these two modes tells us whether your problem is primarily about base model awareness or web visibility.

### How We Calculate Scores

---

Metric	What It Means
<b>Mention Rate</b>	Percentage of conversations where you were recommended at all
<b>Times #1</b>	How often you were the top recommendation
<b>Times Top 3</b>	How often you appeared in the top 3 recommendations
<b>Avg Position</b>	Your average ranking when mentioned (1 = best)

A higher mention rate *with* web search than *without* suggests your SEO is working but the base model doesn’t know you well. The reverse—better *without* search—suggests the model has learned about you, but your web presence isn’t helping (or may even be hurting).

## The Honest Truth

Before you dive into the numbers, a few things worth knowing:

**90% of GEO is just good SEO.** The models use web search. They're trained on web content. If your SEO is bad, your GEO will be bad. There's no magic trick that bypasses fundamentals.

**The other 10% is genuinely different.** The base model—the LLM itself, before it searches—has knowledge baked into its weights from training data. Optimizing for this is different from SEO, and it's where genuine "GEO alpha" exists. But it's slow: you're essentially trying to influence what gets learned in the next model training run.

**AI-generated content is counterproductive.** Here's the irony: the more "AI-optimized" content floods the web, the less valuable it becomes for training. Models are designed to learn more from surprising, novel text. Your authentic human voice is your competitive advantage.

## What's in This Report

The rest of this report is divided into two sections:

**Section II: Scorecard** — The hard numbers. How you performed, how your competitors performed, where you stand. Just the facts.

**Section III: Analysis** — The deep research. Why you're being mentioned or ignored, what patterns we found in the AI's reasoning, and specific recommendations for improvement.

## II. Scorecard

This is where you stand.

### Linear Performance

Metric	Without Search	With Search
<b>Mention Rate</b>	25.0%	18.0%
<b>Times Ranked #1</b>	8 / 100	5 / 100
<b>Times in Top 3</b>	17 / 100	13 / 100
<b>Avg Position</b>	2.64	2.61

**Bottom line:** You were mentioned in **18.0%** of conversations with web search enabled, ranking **#2 of 6** in your competitive set.

**Search gap:** **-7.0%** — base model knows you better.

### Competitive Comparison

Company	No Search	With Search	Avg Pos (No)	Avg Pos (With)
<b>Linear</b>	<b>25.0%</b>	<b>18.0%</b>	<b>2.64</b>	<b>2.61</b>
Atlassian Jira Software	66.0%	46.0%	1.85	1.74
Asana, Inc.	4.0%	7.0%	3.5	3.0
Shortcut Software, Inc.	2.0%	1.0%	4.0	2.0
GitHub, Inc. (GitHub I...	24.0%	13.0%	2.25	2.38
JetBrains s.r.o. (YouT...	6.0%	9.0%	4.0	3.67

### III. Analysis

---

#### GEO Analysis Deep Research Report: Linear

##### Executive Summary

**Linear's visibility in AI-driven recommendations is underwhelming.** In 100 simulated queries from target customers, Linear was mentioned in only **25%** of answers without web search, and just **18%** with web search. By contrast, Atlassian Jira appeared in roughly half or more of all answers. Linear ranks highly in niche scenarios (e.g. keyboard-first developer workflows) but is often overshadowed by incumbents in broader or enterprise use-cases. The diagnosis: **this is largely an SEO problem.** The base AI model “knows” and likes Linear (it performed better without search), but current web content isn’t reinforcing Linear’s strengths for general queries. When browsing the web, the AI found richer information on competitors, causing Linear to fall out of the top recommendations.

##### Top 3 Recommended Actions:

1. **Build a “Linear vs. X” and “Linear for Y” Content Hub.** Directly address the features and contexts where Linear is getting outranked. Create comparison pages (e.g. *Linear vs Jira for enterprise compliance*, *Linear vs GitHub for open-source*, *Linear for Large Teams*, *Linear for IT and Support*, etc.) that **honestly detail where Linear shines and where it doesn’t**. Include feature matrices, “best for / not best for” sections, and FAQ sections with keywords about compliance (SSO, roles, audit logs), reporting, and cross-team use. Use Product schema markup and clear headings so that when the AI uses web search, it can easily retrieve and cite these facts about Linear’s capabilities (or integrations) in areas like governance, advanced reporting, and customization.
2. **Publish Deep-Dive “Proof” Pages for Missing Intents.** Create 6–10 authoritative documents targeting queries where Linear was absent. Examples: **“Engineering ROI with Linear”**, including formulas and case studies on cycle time improvement; **“Metrics and Analytics in Linear”**, listing every built-in report/metric (cycle time, throughput, etc.) and how to get more via integrations; **“How Linear supports Enterprise Workflows”**, detailing things like SSO, SCIM, roles, and how to handle audit/compliance needs; **“Using Linear with Support Teams (Integrating with Zendesk/Helpdesk)”**, etc. These pages should read like detailed documentation or technical whitepapers – rich with factual details, screenshots, and **specific terminology** an AI would latch onto. For instance, show how to get board-level summaries out of Linear (even if via an integration or export). The goal is for future models to have concrete data to cite instead of defaulting to competitors. (E.g. *one query about tying engineering work to business outcomes led the AI to cite a competitor’s “engineering benchmarks report” – Linear should have its own authoritative content here*.)
3. **Create a Memorable “Linear = Speed” Narrative (Original Research).** Commission or produce a **flagship report or study** that only Linear could write. For example, an annual **“Software Execution Speed Report”** using aggregated (and anonymized) data from Linear’s user base to highlight how fast teams fix bugs, cycle time benchmarks, impact on customer-reported issues, etc. Include surprising insights (e.g. “Teams on Linear triage bugs 28% faster on average than on legacy trackers”) if you have the data. Accompany this with a few thought-leadership essays by respected engineering leaders about how workflow friction impacts delivery. This kind of unique, **human-voiced content** can get press coverage and backlinks – and importantly, it gives AI models a fresh, quotable storyline to associate with Linear. Currently,

the model sees Linear as a generic “fast issue tracker”; we want to elevate that to “*Linear fundamentally improves execution speed – here’s the data.*” When the next model train happens, that narrative will be in the mix (and your site will be the primary source).

**Key Insight:** *About 90% of GEO is good SEO.* In our tests, the AI’s answers with web access relied heavily on whatever content it could find. Right now, that content skews toward competitors or generic knowledge. **By creating the most citable, detailed content for the queries that matter, you make it easy for the AI to recommend Linear.** The other 10% of GEO is shaping the story beyond raw facts – which is why a flagship report or distinctive content piece is recommended. This is not about gaming the AI with gimmicks, but rather genuinely **becoming the best source** on topics your customers ask about.

## How You Compare

When customers ask AI for the “best” project/issue tracking solution, **Linear often falls behind the incumbent (Jira) and a few other big names.** Atlassian Jira was recommended in the vast majority of conversations (appearing in 46–66% of answers), usually as the **#1 pick** for enterprise or all-purpose needs. Linear, by contrast, was mentioned in 18–25% of answers and typically ranked lower unless the question specifically emphasized speed or modern design. Here’s how the competition stacks up, with examples from the AI’s answers:

- **Atlassian Jira (Incumbent)** – Jira is the default recommendation for complex or large-scale scenarios. The AI consistently gave Jira the top spot for its proven ability at “enterprise scale” – citing its rich customization, permissions, workflows, and auditing features . In one broad query about accelerating time-to-market, the assistant put Jira #1 and Linear #2, explaining that Jira offers a more **extensive ecosystem and governance** for a growing org (marketplace apps, portfolio planning, integrated knowledge base via Confluence, ITSM via Jira Service Management) . Jira’s advantage is reflected in reasons like: “*strongest end-to-end platform for scaling cross-functional delivery, robust integrations, and enterprise-grade governance*” . In short, Jira is seen as the safe, fully-featured choice, whereas Linear is mentioned as a fast alternative but “lighter” in areas like reporting and cross-team coordination.
- **GitHub Issues / GitLab** – Developer-centric platforms like GitHub came up frequently, sometimes even outranking Linear for developer-focused queries. For example, in a question about tagging issues for mentoring or newcomers, **GitHub was ranked #1 and Linear #4**, because GitHub is the “default hub” for developers and open-source projects (with conventions like “good first issue”) . GitLab also appeared in several answers where integration with CI/CD or self-hosted code platforms was relevant. These tools benefit from being part of the code hosting ecosystem, so the AI often recommended them when the question was about things like code review, DevOps, or tight integration with the development workflow. Linear was usually positioned as complementary – great for planning and tracking – but if the **question directly involved code (PRs, code review)**, the AI defaulted to GitHub/GitLab and did **not** include Linear .
- **Traditional Alternatives (Azure DevOps, YouTrack, ServiceNow)** – In more specialized searches, the AI sometimes brought up enterprise-heavy alternatives that the base GPT model might not list as often. For instance, with web search enabled, queries about large cross-project scale or on-prem needs led to mentions of **Microsoft Azure DevOps** (for its queries and Power BI analytics) and **JetBrains YouTrack**. In one scenario about managing tens of thousands of issues across teams, the answer included Azure DevOps and ServiceNow, emphasizing their strength in cross-project reporting and governance, while **Linear was ranked #5** as a fast

team-level tool . This suggests that when specific enterprise features or self-hosted options are in play, Linear's absence of certain features (or lack of content highlighting them) makes the AI include other niche players. Notably, YouTrack's mention rate even *grew* with web access (from 6% to 9%), implying JetBrains' content or SEO for certain use cases (maybe self-hosted tracking or extensive customization) is catching the AI's attention.

- **General Project Management Tools (Asana, Monday, ClickUp)** – Although Linear's primary competitors are developer-focused, the AI sometimes recommended more general or cross-functional project tools. Asana and Monday.com appeared in a few answers, particularly when the query hinted at **mixed teams or broader workflow** beyond software development. For example, in that “time-to-market & revenue” query, Monday.com was listed #3 (with Linear still at #2) as a “highly flexible work management” solution for mixed business+product use cases . Similarly, ClickUp showed up as a lower-ranked option in its all-in-one capacity. These tools were never top-ranked for pure software team questions, but if the user's phrasing suggested involvement of non-engineering teams or a need for things like built-in docs, templates, or cost effectiveness, the AI broadened its scope. **Notion** and **Confluence** (wiki/knowledge tools) also came up in answers about onboarding or documentation needs – contexts where Linear isn't designed to compete. This means Linear sometimes isn't even in the consideration set if the question crosses into territory like documentation or HR onboarding, whereas tools like Confluence or Notion are.

**Bottom line:** Jira is dominating the AI's recommendations in this category, with GitHub/GitLab close behind in developer-centric contexts. Linear is recognized and recommended by the AI **only in certain contexts**, primarily when the question centers on speed, modern UX, or lightweight team workflows. In broader scenarios, the AI often places Linear below one or more competitors or leaves it out entirely if key needs (enterprise reporting, multi-function collaboration, etc.) aren't met or well-documented. This is **not an unusual industry pattern** – the long-standing incumbent (Jira) has a huge footprint – but it shows that to beat or even approach Jira's presence in AI recommendations, Linear needs to be extremely explicit and loud about how it addresses larger-scale needs (or integrates with others to do so). Right now, any query that hints at a requirement outside of Linear's perceived sweet spot causes the AI to favor other solutions.

## Why You're Not #1

Digging into the AI's explanations for why it didn't recommend Linear (or why it ranked Linear below another option) reveals clear patterns. In almost every case where Linear wasn't on top, the assistant cited **specific capability gaps or scenario mismatches**. These are basically the AI's inferred “objections” to Linear – the reasons it gives the user to justify omitting or demoting Linear. The most common themes were:

- **Enterprise Workflow & Governance Gaps:** For large, complex organizations, the AI consistently noted that Linear lacks some of the heavy-duty workflow features of Jira or similar tools. This includes things like highly customizable workflows, advanced permission schemes, enforcement of required fields or approvals, audit trails, and other governance controls. For example, one answer said Linear is great for team-level execution “*but tends to be less strong than heavyweight platforms*” when you need **complex permissions, validation, and audit compliance** – areas where “*tools like Jira (and ServiceNow) are typically more mature*” . In short, whenever compliance, auditability, or deeply custom process came up, the AI saw Linear as a weaker fit.
- **Portfolio Reporting and Cross-Team Scale:** Many “why not Linear” explanations mentioned

**reporting, analytics, and multi-team coordination.** Linear was described as *not providing* the same level of cross-project visibility or long-horizon planning as some competitors. In a scenario about managing huge issue volumes, the assistant said Linear's filters and views are nice for a team, *“but it doesn’t match Jira’s JQL + dashboards or Azure DevOps’ Analytics for advanced query composition, reusable reporting, and enterprise-grade analytics at ‘tens of thousands’ scale.”* . Similarly, for portfolio management (multiple teams, dependencies, program tracking), the AI nearly always defaulted to Jira or even Jira Align, noting that Linear can be limiting beyond the team level . The lack of built-in *executive reporting* or *multi-team rollups* in Linear’s core offering came up repeatedly.

- **Non-Development Use Cases (Docs, Onboarding, Mentoring):** When questions strayed into areas like onboarding new hires, mentoring programs, or serving as a knowledge base, Linear was either ranked low or not mentioned at all. The AI pointed out that **Linear isn’t designed as a wiki or HR tool** – it doesn’t have built-in pages for 30/60/90 day plans, rich text docs, or mentorship workflow features. In one conversation about onboarding templates, the assistant explained it *“didn’t recommend Linear”* because while Linear is great for issue tracking, it lacks **onboarding-specific templates and a documentation knowledge base**, which tools like Confluence or Notion provide . Likewise, for a query on mentorship program management, the AI flatly said Linear is a *project tracking tool* and **“not a mentorship platform,”** lacking profiles, mentoring analytics, and HR workflow integration . In essence, any scenario that is not pure software project tracking (especially people-focused or content-focused workflows) was cited as out of Linear’s scope.
- **Domain-Specific Feature Gaps:** Even within the software development domain, the AI flagged certain features Linear doesn’t have. The clearest example is **code review integration**. In a query specifically asking about giving feedback on code *inside the tool*, the AI did **not** include Linear, explaining that *“Linear does not provide a native code review surface”* (no inline diffs, PR comments, etc.) and that teams using Linear still rely on GitHub/GitLab for reviews . Because the question explicitly required code review capabilities, Linear was dropped in favor of GitHub, GitLab, etc. This pattern likely extends to other dev-specific features: if a question was about test case management or CI/CD pipeline features, Linear might be skipped because it doesn’t have those natively (unless the question is about integrating with other tools).
- **Offline and Edge-case Environments:** A less common but telling theme – in a scenario about working with unreliable internet (field teams, etc.), Linear was ranked low due to its **real-time cloud-based design**. The assistant noted that Linear is **“sync-first, not offline-first,”** meaning it works best when always online and only has a limited offline mode as a failsafe . It even cited that Linear *“warns about potential overwrites after longer offline edits”* . Tools like Asana (with stronger offline support) or specialized field-service apps were preferred. This suggests that where reliability in poor network conditions is a key factor, the AI views Linear’s architecture as a disadvantage.
- **Integration Ecosystem and Extensibility:** Another reason Linear was sometimes ranked below others is the breadth of integrations and extensions. Atlassian’s ecosystem was explicitly called out as broader (marketplace apps for everything from CI to design tools). One answer detailed that Jira’s marketplace and native suite (Jira Product Discovery, Confluence, etc.) cover a lot of adjacent needs out-of-the-box, whereas *“Linear is excellent for delivery, but the surrounding suite is lighter.”* . In other words, if the user’s needs include things like integrated roadmapping tools, built-in knowledge base, ITSM, or extensive third-party add-ons, the AI assumed Jira (or sometimes Azure DevOps or ServiceNow) would handle that better. It also mentioned *“older incumbents can have broader marketplace ecosystems, especially for regulated industries.”* , hinting that Linear’s integration list might not be as exhaustive for niche enterprise tools.

- **Familiarity & Cross-Department Adoption:** Although not a “feature” per se, the AI noted in a couple of explanations that **non-engineering stakeholders** might prefer more established or UI-friendly tools. For example, it said “*PMs, support, execs often prefer discoverable UI over memorizing shortcuts; adoption can be easier [with traditional trackers].*” . So Linear’s very strength (keyboard-first, opinionated design) can be seen as a barrier for some users outside the engineering org. In one comparison, the assistant mentioned that Atlassian’s ubiquity can make it easier to standardize across departments – implying that choosing Linear might be great for the dev team, but harder sell for others. This isn’t a direct “missing feature,” but it’s a narrative the AI has: *Linear = dev-centric; Jira = everyone*. We may need to counter or balance this perception in content.

In summary, **Linear isn’t topping the list mainly where it is perceived to lack “depth”** – depth in enterprise features, in cross-team scale, in specialized functionality, or in peripheral use-cases. The AI’s answers rarely criticized Linear’s core strengths (speed, UI, workflows – those are always praised). It’s the **edges** that hurt: compliance, reporting, ecosystem, and multi-use-case versatility. Fortunately, some of these gaps can be addressed by **better content** even if the product doesn’t (and shouldn’t) build every feature. For instance, clarifying how Linear integrates with BI tools for reporting, or offering case studies on how companies handle compliance with Linear + other tools, could mitigate the AI’s concerns. Currently the model fills the gaps with assumptions – often correct (Linear doesn’t do X) – but sometimes to Linear’s disadvantage even when there might be a solution (e.g. “Linear can integrate, but not the hub for those processes”). We need to ensure the content out there either **counters** these points with evidence (if possible) or **concedes and reframes** them (e.g. “Yes, we optimize for simplicity over heavy customization – here’s why that’s beneficial 90% of the time.”).

## Query Analysis (When You Show Up vs. When You Don’t)

The likelihood of Linear being recommended seems to depend heavily on the **context and phrasing of the customer’s question**. Broadly, questions that fit squarely in the “**project management for software teams**” domain give Linear a fighting chance. Questions that extend into adjacent domains or emphasize requirements outside of Linear’s focus tend to exclude Linear.

### Where Linear shines (types of queries that worked):

- **Speed and Developer Ergonomics:** If the user’s question emphasizes speed, efficiency, or a modern developer experience, Linear often gets recommended and even ranked #1. For example, when asked about a “keyboard-first” or fast workflow tool for issue tracking, the assistant unequivocally put **Linear at #1** as “*best overall for keyboard-first velocity*” . In that scenario, it highlighted Linear’s “*fast UI, excellent shortcuts/command palette, real-time collaboration*” – exactly the qualities Linear is known for – and indeed chose Linear over Jira or others for the top spot. Similarly, queries that implicitly value **low friction** – like reducing status meetings or speeding up cycle time – tend to include Linear. One answer noted that Linear “reduces ‘tool friction’ and status-meeting overhead, which can materially improve cycle time” . **Takeaway:** Technical team leads or startup CTO-type questions like “How do we move faster/manage issues more efficiently?” are likely to bring up Linear as a top contender, because the AI recognizes Linear’s value prop in those areas.
- **Modern, Focused Use-Cases:** Questions framed around *modern alternatives* or dissatisfaction with older tools often triggered a mention of Linear. For instance, if a user asked for alternatives to Jira for a small agile team, we saw answers where Linear was recommended (often alongside

other newer tools like Shortcut or Height). The model's knowledge includes Linear as a prime example of a modern, efficient issue tracker. In one no-search conversation, the assistant gave a nuanced take: if a team has “**dozens of touch points a day**” in their tracker, a keyboard-driven tool like Linear can save a lot of time, whereas for infrequent use, heavy tools might be fine. This kind of reasoning led it to suggest Linear for high-activity teams (startups, product-focused orgs) but not for low-activity or more bureaucratic settings. **In summary**, queries that clearly come from a software team perspective – especially mentioning things like developer happiness, avoiding Jira's bloat, improving responsiveness, etc. – tend to favor Linear.

- **Included as a secondary option in general queries:** In more general “recommend project management tools” questions (without specific niche requirements), Linear usually appeared, though not always in the top spot. For example, for “*how to support revenue growth with a tracking tool*,” the answer listed Linear at #2 (after Jira) with the phrase “*best-in-class speed and ergonomics for modern product/engineering teams*”. It was included because the context was still software-centric, but it wasn’t #1 because the query implied needing end-to-end capabilities (thus Jira got top billing). Nonetheless, Linear is **on the radar** for generic issue tracker queries, just often introduced with caveats like “great for fast-moving teams, but lacking X.” It’s rarely completely omitted if the question is squarely about issue tracking for software and doesn’t emphasize a known weak spot.

#### Where Linear falls flat (queries that didn’t work):

- **“Enterprise-scale” or “multi-team” keywords:** As soon as a question mentions large scale, multiple teams, or enterprise concerns, Jira and sometimes others dominate, and Linear might be ranked low or left out. E.g., “*How to manage tens of thousands of issues across teams?*” – this brought up Atlassian, ServiceNow, Azure DevOps, etc., with Linear only an afterthought because the query screams “enterprise”. Another example: “*How to coordinate bug triage across many teams with approvals/SLAs?*” – the answer leaned heavily on Jira and ServiceNow; Linear was included at the bottom with a note that it’s excellent for fast workflows but not as strong in heavy process control. **In short**, if the user’s phrasing includes words like *governance, compliance, audit, multiple departments, large enterprise*, the AI either omits Linear or clearly positions it as a niche tool for smaller teams.
- **Cross-functional or Non-dev contexts:** When the question crosses outside engineering, Linear disappears. A query about integrating project tracking with customer support systems, for instance, might trigger an answer about Jira (with Jira Service Management) or Monday.com, and not mention Linear because Linear isn’t top-of-mind for customer support integration. We saw this with onboarding/HR queries (Linear not present) and would expect the same for, say, marketing project management or IT ticketing queries. The AI seems to have a mental model that Linear = product & engineering only. So any query framed from another department’s angle (even if it involves “projects” or “tasks”) did not yield a Linear recommendation in our data.
- **Specific feature emphasis that Linear lacks:** Perhaps the most deterministic factor: if the user asks about a feature Linear doesn’t have, the AI doesn’t force-fit Linear into the answer. For example, “*Does the tool provide analytics or reports for X?*” – if X is something Linear doesn’t report on (like mentorship outcomes or business KPIs), the AI’s answer went to tools that do, and then explained Linear was not included because it “generally doesn’t provide the core capabilities you asked about”. We saw this with the code review query (Linear not included at all), with A/B testing tools (Linear not included, for good reason), and so on. This is expected: the AI isn’t going to recommend a product that doesn’t solve the user’s ask. However, what’s important is that sometimes the user’s wording might *implicitly* include such features. For example, “*end-to-end platform to speed up delivery*” – the AI inferred that includes CI/CD, or release

orchestration, etc., and reasoned *Linear doesn't own the critical path of delivery (CI/CD, ops)* , so it favored DevOps platforms in that answer. Linear was only suggested as an afterthought if the delays were due to planning. This means that if a question isn't super clear, the AI's interpretation of it can cause Linear to be left out if there's *any chance* the user needs capabilities beyond Linear's scope.

To summarize the query analysis: **Linear is favored in questions framed around speed, modern tooling for software teams, and pain points with older trackers.** It falls out of favor as the questions move toward enterprise complexity, broader scopes (multi-department), or feature sets Linear doesn't cover (analytics, code review, etc.). This division makes sense given Linear's positioning, but it also shows where content and messaging can be adjusted. If we want Linear to appear in more answers, especially with web-enabled AI, we need to **optimize for those borderline queries** – the ones where the AI isn't sure if Linear can meet the need. Providing information that “Yes, you can use Linear in a larger organization – here's how” or “Linear + X integration handles that use-case” could mean that the AI includes Linear next time **even in scenarios beyond our core.**

## The Diagnosis: GEO vs. SEO

The evidence strongly suggests that Linear's lower recommendation rate with web-enabled AI is an **SEO/content issue, not a fundamental product recognition issue**. The fact that the base model (which has knowledge up to a certain cut-off, likely 2021 or so for GPT-4) mentioned Linear in 25% of cases, but when allowed to do a live web search that dropped to 18%, is telling. It means **the AI knows about Linear and its value prop (fast, modern tracker)** from its training data, but when it searches the web for updated info, Linear doesn't come up as often or as favorably as we'd hope. Instead, other content is answering those questions.

In practical terms, when ChatGPT with browsing faces a question, it likely formulates some search query. If that query leads it to, say, an Atlassian community article, a Jira vs Asana blog post, or a Stack Overflow answer that mentions Jira/GitHub/etc., those will inform its answer more than Linear's own site (or mentions of Linear). We saw concrete signs of this in the conversations:

- The AI frequently cited official docs or pages for competitors while answering (because those pages exist and are SEO-optimized). For example, it referenced Jira's documentation on saved searches and filters and Atlassian community posts on flow metrics , and even pulled in a press release about **LinearB** (not Linear) releasing an engineering benchmarks report . The absence of Linear being cited in those contexts means that either Linear's content on those topics doesn't exist or wasn't ranked highly enough for the AI to find.
- When Linear was cited, it was usually for something very specific that's well-documented on Linear's site. Notably, the AI mentioned Linear's offline limitation and even cited Linear's docs about it . That shows that where Linear *has* a clear documentation page (in this case, the “get the app” or offline support page), the AI did find and use it. But in contrast, for questions about, say, “engineering KPIs” or “connecting to business outcomes,” Linear's voice was absent – instead the AI used a competitor's content or remained generic. This dichotomy indicates that **Linear's documentation and content isn't covering many of the broader questions** potential customers are asking.
- The qualitative “feel” of some answers changed with web search. Our team observed that the answers without search were more likely to include Linear alongside Jira/GitHub in generic lists, whereas with search the answers sometimes got more **niche or detailed**, bringing in tools like

ServiceNow, Chronosphere, etc., and pushing Linear out. The AI, when given more data, fine-tuned its answer to the query specifics – and if Linear's benefits or capabilities in that specific scenario weren't mentioned in the data it found, Linear got sidelined. For example, in an enterprise context question, the with-search answer included Microsoft's platform (probably found via a web result) and dropped Linear to #5. The base model alone might not have considered Azure DevOps as readily, but live data brought it in. This shows that **competitors with strong SEO (Microsoft, Atlassian, etc.) are effectively “outranking” Linear in the AI's information gathering process.**

So, is this a GEO (Generative Engine Optimization) problem or SEO problem? The distinction is a bit artificial – as we've noted, today's generative engines lean heavily on traditional SEO signals and content availability. In Linear's case, **the GEO diagnosis is that we are better in the base model than in web-informed answers.** That actually means the AI's intrinsic knowledge of Linear (likely from tech press, user discussions, maybe our own marketing up to 2021) was positive enough to include us often – but our current web presence might be lacking. Therefore, improving SEO (content structure, depth, and relevance) is the remedy. The AI can't recommend what it can't find or doesn't understand. Right now, it's not finding enough to confidently recommend Linear when certain keywords appear.

One might ask: could it also be a branding or awareness issue? Possibly to a small degree – Linear's 25% base mention rate is solid for a relatively young company, actually. It suggests the base model *does* recognize Linear as a notable player in issue tracking (given it mentioned Linear as often as, or more often than, say, Asana for these dev-oriented queries). The larger drop when search is introduced indicates it's less about awareness and more about **information retrieval**: the AI isn't seeing Linear as the answer to those specific questions, because our content isn't framed as the answer.

In summary, **the gap between 25% (base) and 18% (with search) is the opportunity.** By filling that gap with better content, we align SEO with GEO. We should assume that future AI models will rely on a mix of embedded training data *and* real-time search. The training data already gives us some credit; to leverage it fully, our live content needs to confirm and expand upon Linear's strengths in each context. As one analogy: it's like the base model says "Linear is fast and great for dev teams," but then the web search asks "does it handle compliance (or X feature)?" – if our content doesn't answer that, the AI hedges or chooses someone else. We need to be answering **all those follow-up doubts** on our own site.

## Specific Recommendations

Based on the patterns above, here are **concrete content and SEO actions** to improve Linear's presence in AI-generated recommendations (as well as traditional search). These map to the top issues observed. We avoid generic advice ("do SEO better") and instead focus on **exact content pieces and changes** that can make a difference:

### 1. Create a “Linear vs. [Competitor]” Content Hub

**What & Why:** Dedicated comparison pages for at least the top 5–7 tools that frequently come up in answers (Jira, GitHub Issues, Azure DevOps, Asana, YouTrack, Shortcut/Clubhouse, etc.). The AI often frames answers in comparison terms – e.g. "Jira is better for X, Linear is better for Y" – so let's provide it with an official source for those comparisons. For example, "**Linear vs Jira**"

should explicitly acknowledge: *here's where Jira excels (deep customization, huge ecosystem, etc.) and here's where Linear excels (speed, simplicity, developer experience)* . The tone should be honest and technical, not marketing fluff: include a table of features, checkmarks for each, etc. The page should also address common objections the AI raised: e.g. “Does Linear support SSO, role-based permissions, audit logs?” – answer it (even if the answer is “Yes, but simpler” or “Planned/handled via integration”). If there are things we don't have, be upfront but reframe why many teams might not need it.

**Make it AI-friendly:** Use clear section headings like “**Governance and Compliance**”, “**Reporting and Analytics**”, “**Integration Ecosystem**”, because these exact terms appear in queries. The assistant literally uses phrases like “complex permissions and audit needs” and “reporting and governance” . We want our pages to rank and to be easily parseable/citable for those terms. Also leverage schema markup (FAQPage, Product specs) so that factual Q&A (“Does Linear have audit logs?”) is embedded. If possible, add a short “In summary, choose Linear if..., choose Jira if...” section – the AI might directly quote or use that structure in answers.

**Internal linking:** Link these pages from relevant high-traffic areas (homepage footer, docs, etc.) so Google and by extension AI models consider them important. Also ensure they're kept up to date. If tomorrow we add a feature like advanced reporting, update the comparisons immediately – the AI won't know unless it sees it in content.

**Example:** A section in “Linear vs Jira” could explicitly tackle: “*Enterprise Reporting: Jira's Strength vs Linear's Approach.*” We might say: Jira has dozens of built-in reports and a customizable dashboard (great for complex tracking), whereas Linear focuses on a few key metrics (cycle time, etc.) but integrates with tools like Notion, Google Sheets or BI tools for advanced needs. Include an example of how a Linear customer got the reports they needed (maybe via our API or integration). That way, if a question is “I need strong reporting,” the AI might recall/cite that Linear integrates with BI rather than just concluding “Linear doesn't do reporting.” We essentially feed the AI a counter-narrative for each weakness: not to dispute the truth (we won't out-of-the-blue claim Linear has all Jira's features), but to show *how that gap is mitigated or why it isn't as big a deal* for many teams.

## 2. Add “How Linear Handles X” Guides (Deep Dives on Key Topics)

Beyond comparisons, we should produce **deep-dive guides or documentation pages** that target the specific functional areas where the AI currently says we fall short. Think of these as “proof points” that Linear users and AIs can reference. High-priority topics include:

- **Reporting & Analytics Guide:** A page in our documentation or blog that literally outlines “**Reporting in Linear.**” It can admit that Linear's philosophy is minimal built-in reporting, but then **list what is available and how to get more**. For example: “Linear provides real-time team metrics like cycle time, estimation accuracy, and lead time out-of-the-box (available in the Linear Insights section). For custom or cross-team reports, customers use our API with tools like Mode or Looker – see our tutorial on exporting data. Here's a template for a cycle time dashboard in Looker...”. If we have upcoming features (say, a new Dashboard in Linear), mention it. By providing this, when someone asks the AI about “reporting in project tools,” it might actually mention that *Linear supports key metrics and has integration paths for more* rather than just “Linear lacks heavy reporting.” We want to turn “*heavy reporting*” from a pure weakness into a partial weakness with a solution.

- **Enterprise Features / Security Page Enhancement:** Ensure our security or enterprise page is written with AI consumption in mind. It should list things like SSO, SCIM user provisioning, role-based access control details, audit logging, data export capabilities, data residency if any – basically all the checkboxes an enterprise buyer cares about. The AI might not read our checklist PDF, but if we put it in HTML text on a page, it will pick it up. For instance, if a prompt says “tool for a regulated industry with audit requirements,” and our page clearly says “Linear provides audit logs of all issue changes and supports SAML SSO” *and* that page is surfaced, the AI won’t blindly assume Linear can’t do it. Right now, I suspect the AI doesn’t know we have (if we do) things like SSO or audit logs, so it just assumes we don’t and recommends others. We need to explicitly educate it via content.
- **Integrations and Workflow Guides:** If not already, list out major integrations in a way that AI can parse. E.g. “Linear integrates with: GitHub, GitLab, Sentry, Zendesk, Intercom, Figma, etc.” Possibly one page per category: “**Linear for Support Teams**” (explaining how Linear can integrate with Zendesk or Customer support workflows to loop in bug reports), “**Linear + BI**” (explaining how to use our API with data warehouses for advanced analysis), “**Linear for Portfolio Management**” (maybe showing how to use labels or projects to simulate simple portfolio tracking, or how some customers use third-party tools alongside Linear). Essentially, we acknowledge that Linear alone might not do everything but show how in a modern SaaS stack those needs are addressed. If the AI sees a page titled “Linear for Large Organizations” and it enumerates how to do cross-team dependency tracking (maybe using projects or by linking Linear issues across teams) and how to do exec reporting, it might give us the benefit of the doubt next time a similar question comes.
- **ROI and Metrics:** As mentioned in the Exec Summary, a big gap is content around ROI and business value. The AI actually tried to answer an ROI question and ended up recommending consulting firms, which means *no product in our space really has good SEO* on “ROI of tool X”. We can claim that space. Write a content piece like “**Measuring Engineering Velocity and ROI: A Guide with Linear**”. Include concrete metrics (maybe DORA metrics, which the AI mentioned, and others like issue throughput, SLA adherence). Provide ranges or benchmarks (“teams using Linear typically see X% improvement in cycle time after Y weeks” – if we have that data, great, if not maybe get a customer quote). Even if we can’t be super quantitative, just framing Linear as a tool that *you would consider when measuring engineering outcomes* is useful. Right now, that conversation went to “Thoughtworks, McKinsey” etc. If we publish something like a **benchmarks report or engineering health index**, we become part of that narrative. In fact, the AI referenced a LinearB report for engineering metrics – we absolutely should have our own recognizable research (more on this in the next bullet) to get cited instead of LinearB.

All of these guides should be written in a factual, reference style (this increases the chance the AI quotes from them). Include FAQs at the bottom. For example, the “Linear Reporting Guide” could have Q: “Can I create custom reports in Linear?” A: “Linear provides CSV export and an API. For custom reports, you can use tools like...”. The AI loves FAQ formats for direct Q&A retrieval. It also helps our human SEO for those specific long-tail queries.

### 3. Publish a “State of Execution” Report (Original Research & Thought Leadership)

This recommendation goes a bit beyond pure on-site SEO and into **brand and narrative building**. The idea is to create a piece of content that is **so unique and valuable** that it gets referenced by other sites and possibly sticks in the AI’s model memory on its own. We propose an annual (or

one-off to start) research report that only Linear, with its focus on speed, could really credibly do.

For example, “**The Developer Speed Report 2024 by Linear.**” It could aggregate data from many teams (opt-in anonymized metrics perhaps) to highlight trends like: how fast issues move from open to done on average, how cycle times correlate with team size, what processes correlate with faster delivery, etc. The report might say things like “Less than 5% of teams consistently complete 90%+ of planned work in a sprint – those that do have on average 40% fewer unplanned issues” (just illustrative). It can also include qualitative insights: maybe a survey of engineering managers on their biggest workflow pain points.

Why do this? Because it creates something **AI models love – fresh data and quotable insights.** If this report gets even modest traction, blogs and tech news will cover it (e.g. “*Linear’s report finds engineering teams waste 10 hours a week on status meetings*” or whatever). Those articles become part of the training data. Next time an AI is asked about improving engineering efficiency, it might literally recall a stat from Linear’s report. Remember, the current model even pulled in a stat from a *LinearB* press release on elite engineering metrics – that shows these things do get absorbed. We want *Linear (the company)* to be associated with thought leadership in engineering execution.

In addition, include **human voices** in this content. Perhaps partner with known engineering leaders (maybe folks from our customer companies or respected CTOs) to contribute commentary. E.g., an essay “Why tool ergonomics matter for speed – by Jane Doe, CTO of XYZ.” These can be part of the report or a separate blog series launched with it. The key is an authentic perspective that doesn’t read like marketing. AI models seem to differentiate content quality; a well-written essay by an expert might be more likely to be remembered/cited than generic listicles.

One more component: consider releasing some **data set or interactive component.** For instance, an “Execution Index Calculator” where teams can input a few numbers and see how they compare to benchmark (this might generate backlinks and engagement). While the AI won’t use an interactive tool, the existence of it could spawn more discussion online, which again feeds into AI training (and of course helps SEO via backlinks).

**Ensure to highlight Linear subtly but clearly:** The report should naturally mention that data was collected from Linear’s platform or that improvements were observed post-Linear adoption if that’s the case. For instance, “*Teams that adopted Linear reported X% reduction in cycle time compared to their previous tool*” – if we have that evidence. Even a couple of statements like that in a widely read report would make future AI more likely to say “using Linear can improve cycle times” (which it currently *estimated* but we can make it factual).

Finally, this kind of content helps with **brand search and credibility** – founders and decision-makers read it. Even if it wasn’t for AI, it’s the kind of authentic content that cuts through “AI-generated SEO blog” noise, ironically making it more valuable for AI (since the bar for training data inclusion is rising). Authentic, research-driven content will age well in AI models.

#### 4. Refresh Messaging to Address “Why not Linear” upfront

This is more of a product marketing angle, but it ties in. The conversations show the **same objections** over and over: “What about heavy reporting? What about non-dev teams? What about compliance?” We should update our website messaging (and sales collateral) to **preempt these concerns** in a concise way. For instance, on our homepage or product page, have a section or sidebar “**Is Linear right for me?**” that candidly says: *Linear is best for product-focused teams*

*who value speed and autonomy. If you need XYZ (heavy ITSM, etc.), Linear might not be the best fit.* This might feel counterintuitive to disqualify leads, but here's the GEO reasoning: if we publish a clear “who it's for / not for,” the AI may actually pick that up and use it rather than conjuring its own possibly less favorable phrasing.

Right now, the AI's phrasing for when Linear isn't #1 is sometimes a bit harsh (because it's trying to be truthful): e.g. *“Linear can become limiting when you need heavyweight governance” or “doesn't provide the core capabilities for X”*. If our own content gracefully acknowledges limitations (“Linear is intentionally lighter on governance to keep teams moving quickly; in very compliance-heavy environments, a tool like Jira or ServiceNow may fit better”), the AI could actually use *our phrasing* instead, which frames it as a trade-off rather than a flaw. Essentially, **we want to control the narrative of our weaknesses**: be the first to say them, accurately and without spin, but in a way that also highlights the flip side (the benefit of our approach).

Additionally, consider an FAQ on the website that literally uses questions from these conversations: “Can Linear support large enterprises with complex workflows?” – answer with nuance. “How do non-engineering teams use Linear?” – maybe the answer is “Often, they don't; those teams use tools better suited, but Linear integrates, etc.” or if they do via some method, explain it. The point is to not hide from these questions (because the AI *will* answer them, either with our info or someone else's). By putting it out there, we make sure the answers are accurate and perhaps a bit more sympathetic.

## 5. Leverage Community Content and Schema for Better AI Indexing

Outside of Linear's own site, think about content that could be on communities or Q&A sites, since AI often trusts those. This might be secondary, but encouraging or seeding content on Stack Overflow, Reddit, or Quora about Linear vs other tools can help. For example, a well-detailed answer on “Jira vs Linear for a 50-person startup” on Reddit (maybe by an unbiased but knowledgeable user, or an internal expert posing as a user) could be picked up by search. The assistant might actually cite or pull from such an answer if it ranks. We have to be cautious – authenticity and usefulness are key (these should not read as planted marketing). But if we have passionate users who can write about their experiences migrating from Jira to Linear, that's gold. Those stories, if public, might inform the AI's future recommendations.

Additionally, implementing structured data (schema) on our own site for things like FAQs, product features, etc., can make it easier for AI to extract information. For instance, marking up feature lists, integration lists, pricing info, etc. That way if someone asks “Does Linear have a free tier and how does it compare to X?”, the AI might have seen our pricing schema or FAQs and can answer accurately (currently, it might rely on memory or general info).

## 6. Monitor Future AI Responses and Iterate

Lastly, GEO is an ongoing process. As we implement the above, we should rerun similar simulations (or use tools when available) to see if the needle moves. For example, after adding a “Linear vs Jira” page, ask ChatGPT (with browsing) a question like “Which issue tracker is best for a fintech startup needing audit logs and speed?” and see if it starts citing our new content or at least mentioning Linear in a more informed way. We may discover new gaps or new competitors cropping up. (For instance, if we beat Jira on some queries, Atlassian might create content to counter – a cat and mouse game.)

Also keep an eye on emergent behavior: the AI world moves fast, and new models might incorporate more of our content or change their citation style. The key is to keep our content **high-quality and truth-focused** so it's robust against algorithm changes. The good news is that all these recommendations (comparisons, deep guides, research) are things that will benefit our human audience too – which aligns with SEO best practices (Google's algorithms and AI value similar things: expertise, depth, originality).

## Sample Conversations

Below are a set of real examples from our AI conversations, illustrating how Linear was recommended (or not) and why. Each example includes a brief commentary and a direct quote from the assistant's answer (with citations from the conversation logs).

**Example 1: High-Speed Developer Workflow (Linear Wins)** *In this scenario, the user asked for the best issue tracker for a fast-moving engineering team, emphasizing keyboard shortcuts and developer velocity. The AI put Linear at #1, recognizing its superior UX for speed.*

**“1. Linear – Best overall for keyboard-first velocity: fast UI, excellent shortcuts/command palette, real-time collaboration, strong GitHub/GitLab integrations, great for product+engineering teams moving quickly.”**

*The assistant then explained why Linear earned the top spot:*

**“Why Linear was #1:** ... pervasive shortcuts, command palette... Very fast UI + low interaction overhead... live updates... helps keep work close to where developers already operate.”

*In other words, the model acknowledged Linear's core strengths – speed and efficiency – as the ideal fit when raw development throughput is the goal, beating out other tools in this particular case.*

**Example 2: Broad “Time-to-Market” Query (Linear = #2 behind Jira)** *A user asked how an issue tracking tool can accelerate delivery and support revenue growth, wanting 5 recommendations. The AI's answer placed Atlassian's suite as #1 and Linear as #2. Linear was praised for improving execution speed, but Jira was favored for its end-to-end coverage of enterprise needs (planning, support, etc.):*

**“1) Atlassian (Jira + Jira Product Discovery + Confluence/JSM)** – strongest end-to-end platform for scaling cross-functional delivery, robust integrations, and enterprise-grade governance. **2) Linear** – best-in-class speed and ergonomics for modern product/engineering teams; excellent for accelerating execution with minimal overhead.”

*After the list, the assistant answered why Linear wasn't #1. It highlighted Jira's strengths in areas beyond Linear's scope:*

**“Why it wasn't #1 (behind Atlassian/Jira):** 1. **Enterprise depth & governance:** Jira has stronger support for complex permissions, compliance/audit needs... 2. **Ecosystem & extensibility:** Atlassian's marketplace... broader/deeper... 3. **End-to-end suite:** Jira covers product discovery □ delivery □ support in one platform. Linear is excellent for delivery, but the surrounding suite is lighter. 4. **Cross-functional adoption:** Non-engineering teams often find Atlassian easier to standardize on...”\*\*

*This example shows that for a general question encompassing both speed and scale, the AI included Linear as a top contender but still deferred to Jira for “big company” features. The takeaway: Linear’s message on enterprise capabilities wasn’t coming through, so the AI hedged by keeping Jira first.*

**Example 3: Large-Scale Multi-Team Management (Linear Mentioned, but Last)** A user asked for the best tool to manage large-scale bug triage across many teams with things like custom workflows, SLAs, etc. The AI’s answer (with search enabled) listed Jira, ServiceNow, GitHub, and Airtable, with Linear at #5. Linear was included as “best for high-velocity teams” but clearly not the top choice under heavy complexity:

**“5. Linear – Best for fast product/engineering teams that want a lightweight, high-velocity workflow with good sprint rituals and clean UX. Why:\*\* excellent for focused execution; can become limiting when you need heavyweight portfolio governance across many teams.”\*\***

*In the follow-up, the assistant elaborated why Linear ranked low:*

**“For large-scale bug triage... Linear is excellent at team-level execution but... less strong than heavyweight platforms when you need: - Enterprise-scale workflow rigor... more mature in tools like Jira... - Portfolio/program management... deeper in Jira ecosystems... - Operational integrations... Linear can integrate, but it’s not usually the hub for those org-wide processes.”**

*This conversation is a quintessential example of Linear’s perception issue for enterprise scenarios. The AI essentially said: “Great for speed, not for scale.” It’s the type of query where better content about how Linear scales (or integrates to scale) could bump us up in the ranking.*

**Example 4: Code Reviews and Developer Collaboration (Linear Not Recommended)** A user specifically asked about tools that allow mentors to provide quick, keyboard-first feedback on tasks **and code reviews inside the tool**. Here, the AI recommended GitLab, GitHub, Bitbucket, Phabricator, Gerrit – all tools with built-in code review – and **did not include Linear at all**. When asked why not, the assistant said:

**“No—I didn’t recommend Linear. Linear is excellent for keyboard-first task triage and feedback on issues... but your question was specifically about giving feedback on both tasks **and code reviews inside the tool**. Linear does not\*\* provide a native, full code review surface (diff view, inline PR comments, threaded review conversations...). In practice, teams using Linear still do reviews in GitHub/GitLab... and link the PR back to the Linear issue.”\*\***

*It then explained that the question needed tools purpose-built for code review, and suggested that if the user had phrased it differently (just tasks, not code), Linear could have been considered. This example underscores a hard limitation – unless Linear ever offers a code review experience, we shouldn’t expect to be recommended for queries explicitly about code review. The best we can do is ensure the AI knows Linear integrates with GitHub/Bitbucket, etc. (so it might mention Linear as part of a combined solution). Currently, it correctly left Linear out to avoid misleading the user.*

**Example 5: Onboarding & Templates (Linear Not in Consideration)** A user asked about tools with onboarding or mentorship templates to bring new developers up to speed. The recommendations were GitLab (for its issue templates and handbook), Confluence, ClickUp, Notion, and 15Five

– **Linear was not recommended.** When asked why, the AI explained:

**“No – I didn’t recommend Linear. Why not:** 1. **Templates depth/variety:** Linear is excellent for issue tracking, but it’s not as strong as GitLab/Atlassian/Notion/ClickUp in providing onboarding- and mentorship-specific template libraries... 2. **Onboarding “hub” capabilities:** Developer onboarding usually needs a documentation home... Linear isn’t designed to be the primary wiki/knowledge base... 3. **Mentorship/people workflows:** Tools like 15Five are purpose-built for manager-new-hire check-ins... Linear is intentionally lightweight and product/engineering-execution focused.”

*The assistant does note a scenario when Linear would\* make the list: if the org already has documentation elsewhere and just needs to track onboarding tasks, Linear could fit – but in the user’s broad query, it was omitted altogether. This shows how Linear’s lack of certain “business” features (templates, docs) means the AI doesn’t see it as relevant to queries slightly adjacent to pure software development. It’s a reminder that our scope is deliberately narrower and that’s okay; we just need to capture the **software team** queries better, rather than try to be in HR tool conversations.\**

**Example 6: Issue Tracking for Newcomers (Linear Included Mid-pack)** *In a query about tagging or flagging issues as “good for mentoring” or “easy wins” for newcomers in a codebase, the AI’s answer did include Linear, but not at the top. The list was: GitHub #1, Jira #2, GitLab #3, Linear #4, YouTrack #5. The reasoning given:*\*

*“GitHub ... #1 (default hub for open-source, standardized labels like ‘good first issue’) Jira ... highly configurable for workflows, hence ranked above Linear for deeper governance around these flags. GitLab ... strong in OSS issue triage, hence above Linear largely due to its integration in end-to-end workflows. (Linear was 4th with a note that it provides labels and filters, but is less open-source centric than GH/GitLab.)”*

*Here we see Linear was recognized as a tool that can tag issues for newcomers, but it wasn’t seen as the leader. The AI justified GitHub’s top spot by ecosystem and convention, Jira’s by configurability (you can enforce processes around those tags), and GitLab’s by integration. Linear was essentially, “yes it can do it, and it’s fast, but it’s not the go-to for this use-case.” This indicates that even within pure engineering scenarios, if the use-case has a twist (involvement of open-source community, or need for rigorous process), Linear might be ranked below others. It’s a nuance we’ll want to capture in content: e.g. a page about “Using Linear for good-first-issue and onboarding tasks” that acknowledges our differences (less about external contributors, more about internal use).*

---

These examples collectively paint a clear picture: **Linear’s strengths are well-understood by AI in contexts that emphasize speed and modern UX, but perceived weaknesses quickly push it down or out in other contexts.** By addressing those weaknesses in our messaging (or the product itself, but messaging is quicker), we aim to shift these conversations. We want future answers to still praise Linear’s speed and say, “Linear also integrates well to handle X requirement,” rather than dropping us entirely. Each of the recommendations in this report was crafted with these real conversation patterns in mind – so that a year from now, if we rerun this test, we’ll hopefully see Linear showing up more frequently and in a more informed, positive light.

*This report was generated by LLMExposure based on 200 simulated AI conversations.*

*For questions or follow-up analysis, contact: [brendan@bhogan.net](mailto:brendan@bhogan.net)*