PM judgment as the moat, eval-driven AI products, and agents as a distribution channel

PM Daily Digest

2026-02-25

PM judgment as the moat, eval-driven AI products, and agents as a distribution channel

By PM Daily Digest • February 25, 2026

This edition focuses on how AI is reshaping product work: PMs' defensible advantage is shifting toward judgment, orchestration, and taste, while AI products demand eval-driven development instead of PRD-driven specs. You'll also find a practical playbook for agent-assisted "vibe PMing," tighter product-to-business alignment, and concrete case studies on evals, prioritization, and pre-PMF growth.

Big Ideas

1) PM value is shifting from "task admin" to orchestrating outcomes—with judgment as the moat

A Group PM at YouTube describes how AI is collapsing execution and administrative burden (e.g., writing PRDs), pushing PMs toward boundary-pushing work, strategic decision-making, and orchestration [1][2]. In this environment, defensibility comes from a "human moat" of:

- **Strategic vision** (AI can't choose the destination) [3]
- **User empathy** (advocating for unspoken human pain beyond pattern-matching) [4]
- **Product taste** (defining what "good" feels like) [5]
- **Communication** (aligning cross-functional teams as silos burn down) [6]

[1] Leading AI Products: Speed & Orchestration | YouTube Group PM
[2] Leading AI Products: Speed & Orchestration | YouTube Group PM
[3] Leading AI Products: Speed & Orchestration | YouTube Group PM
[4] Leading AI Products: Speed & Orchestration | YouTube Group PM
[5] Leading AI Products: Speed & Orchestration | YouTube Group PM
[6] Leading AI Products: Speed & Orchestration | YouTube Group PM

- **Judgment** (applying context/ethics to probabilistic outputs) [7]

This also changes cadence: "months of discovery" compress into "minutes of decision making," with **decision velocity** framed as competitive advantage [8].

Why it matters: As execution becomes cheaper/faster, the bottleneck becomes *what's worth building* and *how fast you can decide* [9].

How to apply: Treat strategic direction, taste, and judgment as first-class deliverables—not "soft skills." Make your week explicitly include decision-making time (not just coordination), and design processes that surface tradeoffs quickly (prototype → evaluate → decide) [10][11].

---

2) For AI products, the core artifact is shifting from PRDs to evals + guardrails

The same YouTube PM argues that to lead AI products, you evolve artifacts from rigid specs to **evals**: you can't write a PRD for how an LLM should "feel or reason," so you encode principles into system prompts and evaluation frameworks, moving from enumerating edge cases to establishing guardrails [12]. This includes explicitly managing a **hallucination budget**—deciding where creativity is a feature vs. a liability—and using better grounding/retrieval (e.g., RAG) where accuracy tolerance is near-zero [13].

[7] Leading AI Products: Speed & Orchestration | YouTube Group PM
[8] Leading AI Products: Speed & Orchestration | YouTube Group PM
[9] Leading AI Products: Speed & Orchestration | YouTube Group PM
[10] Leading AI Products: Speed & Orchestration | YouTube Group PM
[11] Leading AI Products: Speed & Orchestration | YouTube Group PM
[12] Leading AI Products: Speed & Orchestration | YouTube Group PM
[13] Leading AI Products: Speed & Orchestration | YouTube Group PM

Leading AI Products: Speed & Orchestration | YouTube Group PM (12:49)

Why it matters: In a probabilistic system, a single "correct answer" often doesn't exist; you're measuring quality across a distribution, and evals become the "quantitative heartbeat" of the product [14].

How to apply: 1. Translate product principles (tone, safety, pedagogy, etc.) into a rubric (scorecard). [15] 2. Use an **auto-rater / LLM-as-judge** with a "teacher model," golden set, and custom rubrics to grade performance across interactions. [16] 3. When swapping models or updating prompts, use the auto-rater and look for confidence intervals—then evolve the rubric as the product evolves. [17]

---

3) "Agents" are emerging as a distribution channel—products need programmatic surfaces to be discoverable

Aakash Gupta frames a shift: agents don't browse marketing sites or onboarding flows; they call your **CLI**, hit your **MCP server**, and read docs programmatically—without those surfaces, your product is "invisible" to them

[14] Leading AI Products: Speed & Orchestration | YouTube Group PM
[15] Leading AI Products: Speed & Orchestration | YouTube Group PM
[16] Leading AI Products: Speed & Orchestration | YouTube Group PM
[17] Leading AI Products: Speed & Orchestration | YouTube Group PM

[18]. He highlights MCP's rapid adoption (97M monthly SDK downloads in 12 months, 10,000+ active servers; multiple major companies adopting it; donated to the Linux Foundation) and compares running an MCP server to running a web server [19].

Why it matters: If competitors ship an MCP server, agent-based workflows (e.g., Cursor sessions, autonomous workflows) can discover and use their product without humans ever visiting a website [20].

How to apply: Make "agent access" a product surface area review: - Do you have a CLI? - Do you expose MCP endpoints? - Are your docs machine-readable and usable programmatically?

---

4) Impact comes from invalidation and subtraction, not just shipping

Two complementary angles landed this week:

- Tony Fadell's rule from Nest: if you can't explain **why it matters** (the reason a real person would care), it doesn't ship—and that rule "killed dozens of features" [^3].
- Run the Business argues that the pivotal PM skill is asking "**Should we build it at all?**" and learning to abandon low-impact deltas fast; they cite that "90% of the time, validation actually means invalidation" [^4][^4][^4].

Why it matters: Faster execution increases the risk of efficiently building things that shouldn't exist in the first place [^4][^4].

How to apply: Treat "why" as a shipping gate and make invalidation explicit: define what data would *disconfirm* the idea, and plan to stop when it shows up [^3][^4].

Tactical Playbook

1) Set up "Vibe PMing" workflows (Claude Code + MCP + skills)

Gupta describes "vibe PMing" as: you describe the problem; an agent pulls data, analyzes charts, synthesizes feedback, drafts the spec, and files the ticket [^5][^5].

Step-by-step setup: 1. **Create a product context repo** in Cursor/Claude Code (PRDs, plans, roadmap notes, specs) as Markdown files; reference them with `@` to pull context without copy/paste. [^5] 2. **Connect MCP**

[18][Karpathy is telling you something most product teams haven't internalized yet.
[19][Karpathy is telling you something most product teams haven't internalized yet.
[20][Karpathy is telling you something most product teams haven't internalized yet.

servers (at minimum analytics + tickets, e.g., Amplitude and Linear). [^5] 3. **Write "skills"** as Markdown (name, when to use, heuristics); invoke them via `/analyze-chart`, `/analyze-feedback`, etc. [^5] 4. **Manage context deliberately:** when you hit ~80–90% context window usage, write a Markdown summary and start a fresh session; keep only relevant MCPs active. [^5]

Two common failure modes to avoid: - Expecting MCP to orchestrate multi-step workflows by itself; MCP connects your AI to external data/actions, but you still need skills/prompts to tell it what to do [^5]. - Loading too many MCPs at once (adds irrelevant tool descriptions; slows/confuses the model) [^5].

---

2) Align product work to business goals with a "metrics one-pager" (and stop over-attributing)

Mind the Product shares a simple alignment exercise: map the top-level business goal and current "mode" (growth vs. cost-cutting), connect your product's contribution, then break it down into acquisition/engagement/retention metrics in a hierarchy/pyramid [^6].

Step-by-step: 1. Draft the one-pager: business goal + "mode" + the key metric that drives the goal (e.g., a marketplace "matching rate" as a revenue driver). [^6] 2. Break down into the supporting input metrics (acquisition, engagement, retention) and explicitly connect the dots from initiative → metric → revenue logic. [^6] 3. Use it as a stakeholder conversation starter (draft solo first if needed, then validate with stakeholders). [^6]

Communication guardrails: - Don't over-index on precise attribution when many teams contribute; the episode calls "attribution fights" unproductive and notes multi-touch journeys can make exact revenue crediting a waste of time [^6]. - Present with *less is more*: focus on the headline, start with numbers, and end with "what we're learning and what we're doing about it" [^6]. - Limit to **2–3 key points** per slide/argument to avoid overload [^6].

---

3) Vibe prototyping without skipping the problem space (and without mistaking usability for PMF)

Dan Olsen frames a recurring trap: solutioning gets so easy with vibe coding that teams skip problem space—and they also confuse usability feedback with product-market-fit feedback [^7].

Step-by-step discipline: 1. Start from the base of the product-market-fit pyramid (target customer → underserved needs → value prop) before you let tools generate features/UX. [^7] 2. Prototype and test, but explicitly separate: - **Usability feedback** (can hide value if UX is poor) - **PMF/value feedback** (would they use it?) [^7] 3. Use richer inputs to get better outputs: "text

+ image" (color palette, style guide, or a photo of a lo-fi wireframe) outperforms text-only prompts [^7]. 4. For concept prototypes, don't burn time on backend/auth; "fake it" with sample data or local storage to avoid rabbit holes [^7].

---

4) If your product relies on third-party data: build boundaries and reduce "PM-as-bug-middleman"

A Reddit PM described an internal product relying on third-party data, getting direct user complaints for both tech bugs and incorrect input data—and feeling pressure to personally verify every edge case [^8][^8][^8][^8]. Teresa Torres and Petra Wille argue PMs shouldn't own bug tracking/tech debt/architecture, and recommend removing the PM as a middleman via dashboards/shared tools/Slack channels, escalating systemic quality issues to engineering leadership [^9][^9][^9].

Step-by-step: 1. Separate issue types in intake: **data wrong at source** vs. **tech bug** (make the distinction visible to stakeholders). [^8][^8] 2. Provide a direct status path (dashboard/shared tool/Slack channel) so bug status doesn't route through Product. [^9] 3. When quality issues recur, escalate to engineering leadership as a **system problem**, not a queue of individual bugs for the PM to manage. [^9][^9]

Case Studies & Lessons

1) AI tutor: evals revealed a "helpful" answer that failed the product goal

In a tutor example, the model answered a physics question correctly and with a friendly tone, scoring 5/5 on clarity and encouragement—but got 1/5 on pedagogy because the rubric explicitly required *not* providing the final numerical answer, instead guiding the student to the next step. The eval pinpointed the mismatch between default LLM helpfulness and desired product behavior ("the struggle is a feature") [21].

Takeaway: If you can't define "good" via deterministic tests, encode product taste into rubrics and let evals make misalignment measurable [22].

---

[21] Leading AI Products: Speed & Orchestration | YouTube Group PM
[22] Leading AI Products: Speed & Orchestration | YouTube Group PM

2) "Loudest customer wins" prioritization: Aranza auto-extracts requests from Slack and ties them to ARR

A PM built Aranza after frustration that roadmap debates skew toward whoever is loudest because nobody has time to read all Slack threads/tickets [^10]. Aranza reads Slack, extracts feature requests, scores them by revenue impact, and shows who asked for what with their ARR; it's early with 10 users [^10].

Takeaway: Even a lightweight "request → account/ARR attribution" view can shift prioritization discussions from anecdotes to structured inputs (especially in noisy channels like Slack) [^10][^10].

---

3) Pre-PMF growth: optimize for learning, wedge, and inclusion—not "more waitlist signups"

In a pre-PMF GTM thread (wedding planning software), advice emphasized that GTM is less about scaling channels and more about finding **who converts and why**—starting with a clear wedge segment [^11]. It also recommended:

- Treat outreach as education to uncover switching triggers [^11]
- Get to a crisp one-sentence value proposition before scaling distribution [^11]
- Use early users' language/feedback to shape positioning [^11]
- Make beta signups feel inclusive (access + feedback loop + early group), not a generic waitlist [^11]
- Prioritize tight feedback loops over content pushing or early marketing hires [^11]

---

4) "First principles workflows" as product strategy: adapt to variability in how teams work

Scott Belsky highlighted trybasis' approach as a rethink of accounting workflows from first principles—adapting to how different teams operate (because every accounting practice works differently) and "liberating practitioners" to focus on client needs [^12].

Takeaway: Workflow products often win by embracing operational variance instead of assuming one canonical process [^12].

Career Corner

1) AI PM interviews skew heavily behavioral—practice accordingly

Aakash Gupta shares a breakdown from helping hundreds of AI PM job seekers:

- **Behavioral interviews: 75%** (Leadership & Drive 40%, AI-specific experience 25%, Values & Culture 10%) [^13]
- **Case interviews: 15%** (Product sense, Product design, Success metrics each 5%) [^13]
- **Technical interviews: 10%** [^13]

How to apply: Allocate practice time proportionally; don't over-optimize for cases at the expense of leadership stories and evidence you've actually done AI PM work [^13].

---

2) Turn interview prep into a feedback loop with an AI interview coach (transcripts → scoring → drills)

Lenny's Newsletter describes a Claude-based "AI job interview coach" aimed at fixing the lack of usable feedback loops in interviews (impostor spiral, blind grind, and practice scarcity) [^14][^14]. It supports:

- Transcript-based analysis: score answers on substance/structure/relevance/credibility/differentiation and produce a "delta sheet" [^14]
- Mock interviews and drills [^14]
- Story bank creation and retrieval drills [^14]

How to apply (setup): install Claude desktop app, download the GitHub project, rename `SKILL.md` to `CLAUDE.md`, open in Claude's "Code" tab, and type `kickoff` [^14].

---

3) "Default to AI" as a career advantage (and ask for access)

Gupta's tactical career advice includes:

- Default to AI for analysis/specs/strategy as a thought partner [^5]
- Spend weekly time reviewing what shipped in the last 7 days (models, agents, MCP integrations) [^5]
- Learn AI-specific frameworks like evals [^5]
- If your org isn't giving you Claude Code/Cursor access, request it [^5]

Tools & Resources

- **Leading AI Products: Speed & Orchestration (Product School, YouTube)** — PM role evolution, human moat, evals/guardrails, and auto-raters: https://www.youtube.com/watch?v=fUJ4rujs0Ao [23][24]

[23] Leading AI Products: Speed & Orchestration | YouTube Group PM
[24] Leading AI Products: Speed & Orchestration | YouTube Group PM

- **Claude Code + Analytics = Vibe PMing (Aakash Gupta, podcast episode)** — end-to-end agent workflows + MCP pitfalls: https://www.news.aakashg.com/p/frank-lee-podcast [^5][^5]
- **(In)Validation: The Pivotal Product Management Skill (Run the Business)** — invalidation as impact, and why "validation" often disproves ideas: https://runthebusiness.substack.com/p/invalidation-the-pivotal-product [^4][^4]
- **How to align product work to business goals (Mind the Product, YouTube)** — metrics one-pager + communication guidance: https://www.youtube.com/watch?v=oENELPjdDwo [^6][^6]
- **Dan Olsen & David Bland: Vibe Coding Advice for Product Teams (YouTube)** — separating usability vs. PMF feedback; prototyping discipline: https://www.youtube.com/watch?v=woHytMhVe-M [^7]
- **The AI-Native PM (free live workshops from Lenny Rachitsky + Maven)** — themes: AI workflows, becoming more technical, product sense & influence; signup: http://bit.ly/ai-native-pm [^15][^15]
- **How to use AI in your next job interview (Lenny's Newsletter)** — AI interview coach system: https://www.lennysnewsletter.com/p/how-to-use-ai-in-your-next-job-interview [^14][^14]

The new distribution channel for software is agents. Agents don't browse your marketing site, watch your demo video, or click through your onboarding flow. They call your CLI. They hit your MCP server. They read your docs programmatically. If none of those surface areas exist, your product is invisible to them.

Look at how fast this moved. MCP went from zero to 97 million monthly SDK downloads in twelve months. 10,000+ active servers. OpenAI, Google DeepMind, Microsoft, and Cloudflare all adopted it. By December 2025, Anthropic donated MCP to the Linux Foundation because the standard had already won. Running an MCP server is now compared to running a web server.

That's the new baseline for product discovery.

85% of enterprises are expected to have AI agents deployed. Those agents need structured, programmatic access to your product. They need CLIs, MCP endpoints, and machine-readable documentation. A beautiful React dashboard is worthless to an agent trying to pull data into a workflow at 3am.

This tells you everything about why Karpathy's framing of CLIs as "legacy" technology is so precise. Legacy means battle-tested, standardized, universally parseable. stdin/stdout, flags, JSON output. The entire Unix philosophy was accidentally designed for AI agents decades before they existed.

Your competitor ships an MCP server and suddenly every Claude Code user, every Cursor session, every autonomous workflow can discover and use their product. No human ever visits the website. No sales call. No onboarding email. The agent just finds the tool and starts using it.

Here's what you need to stay ahead of this shift:

1. MCP explainer: %5Bhttps://www.youtube.com/watch?v=a9wO6GSAoGk%5D(https://www.youtube.com/watch?v=a9wO6GSAoGk)
2. AI agents guide: %5Bhttps://www.news.aakashg.com/p/ai-agents-pms%5D(https://www.news.aakashg.com/p/ai-agents-pms)
3. Context engineering guide: %5Bhttps://www.news.aakashg.com/p/context-engineering%5D(https://www.news.aakashg.com/p/context-engineering)
4. How to build AI products: %5Bhttps://www.news.aakashg.com/p/how-to-build-ai-products%5D(https://www.news.aakashg.com/p/how-to-build-ai-products)
5. Claude Code guide: %5Bhttps://www.news.aakashg.com/p/claude-code-v21-is-insane-ai-update%5D(https://www.news.aakashg.com/p/claude-code-v21-is-insane-ai-update)

The companies that win the next 24 months are the ones building agent-accessible surface area right now. The ones that lose are still optimizing their landing page above the fold.

Andrej Karpathy
@karpathy

CLIs are super exciting precisely because they are a "legacy" technology, which means AI agents can natively and easily use them, combine them, interact with them via the entire terminal toolkit.

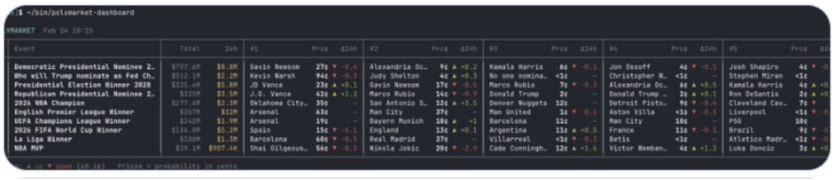
E.g ask your Claude/Codex agent to install this new Polymarket CLI and ask for any arbitrary dashboards or interfaces or logic. The agents will build it for you. Install the Github CLI too and you can ask them to navigate the repo, see issues, PRs, discussions, even the code itself.

Example: Claude built this terminal dashboard in ~3 minutes, of the highest volume polymarkets and the 24hr change. Or you can make it a web app or whatever you want. Even more powerful when you use it as a module of bigger pipelines.

If you have any kind of product or service think: can agents access and use them?

- are your legacy docs (for humans) at least exportable in markdown?
- have you written Skills for your product?
- can your product/service be usable via CLI? Or MCP?
- ...

It's 2026. Build. For. Agents.



](https://substack.com/@aakas

219158633) [^3]: post by @tfadell [^4]: (In)Validation: The Pivotal Product Management Skill [^5]: Claude Code + Analytics = Vibe PMing [^6]: How to align product work to business goals | Corinna Stukan (CEO, Bizzy) [^7]: Dan Olsen & David Bland: Vibe Coding Advice for Product Teams [^8]: r/ProductManagement post by u/YesterdayDreamer [^9]: post by @ttorres [^10]: r/ProductManagement post by u/Gautamagarwal75 [^11]: r/startups comment by u/Vast_Deal_7114 [^12]: post by @scottbelsky [^13]: 75% of AI PM interviews are behavioral. [^14]: How to use AI for your next job interview [^15]: post by @lennysan

---

Sources

1. Leading AI Products: Speed & Orchestration | YouTube Group PM
2. [Karpathy is telling you something most product teams haven't internalized yet.

The new distribution channel for software is agents. Agents don't browse your marketing site, watch your demo video, or click through your onboarding flow. They call your CLI. They hit your MCP server. They read your docs programmatically. If none of those surface areas exist, your product is invisible to them.

Look at how fast this moved. MCP went from zero to 97 million monthly SDK downloads in twelve months. 10,000+ active servers. OpenAI, Google DeepMind, Microsoft, and Cloudflare all adopted it. By December 2025, Anthropic donated MCP to the Linux Foundation because the standard had already won. Running an MCP server is now compared to running a web server.

That's the new baseline for product discovery.

85% of enterprises are expected to have AI agents deployed. Those agents need structured, programmatic access to your product. They need CLIs, MCP endpoints, and machine-readable documentation. A beautiful React dashboard is worthless to an agent trying to pull data into a workflow at 3am.

This tells you everything about why Karpathy's framing of CLIs as "legacy" technology is so precise. Legacy means battle-tested, standardized, universally parseable. stdin/stdout, flags, JSON output. The entire Unix philosophy was accidentally designed for AI agents decades before they existed.

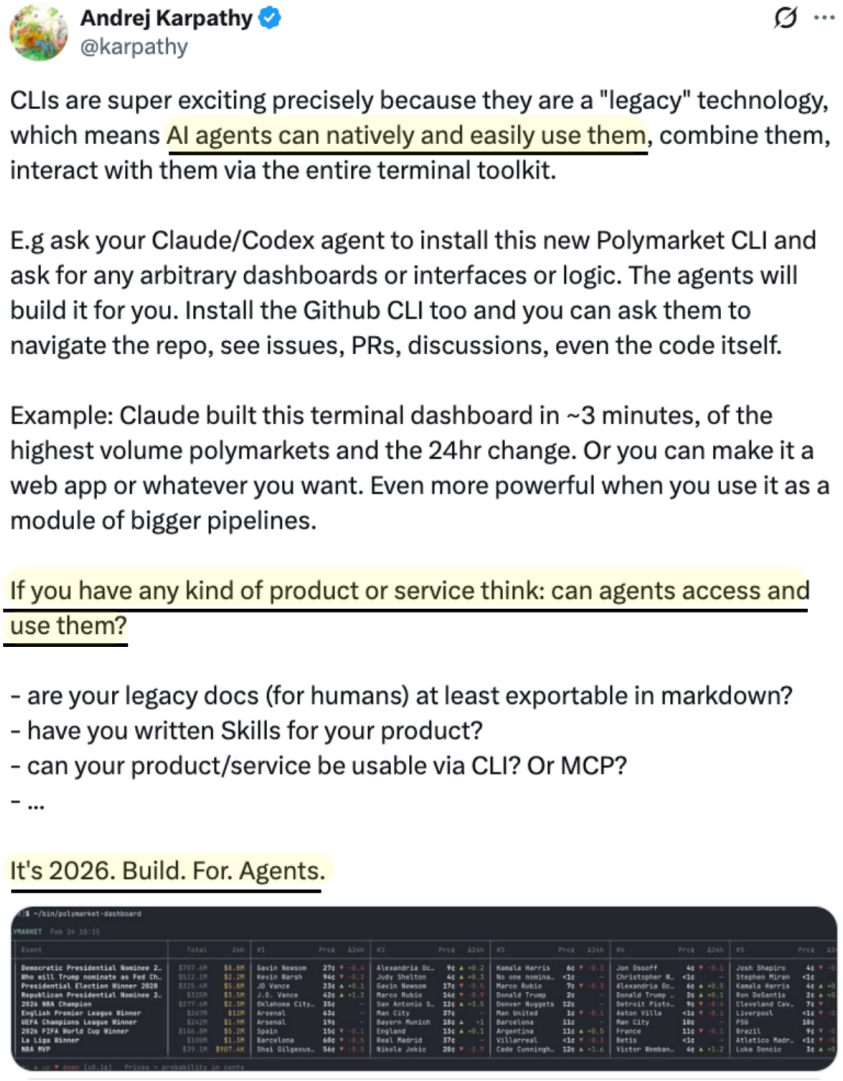
Your competitor ships an MCP server and suddenly every Claude Code user, every Cursor session, every autonomous workflow can discover and use their product. No human ever visits the website. No sales call. No onboarding email. The agent just finds the tool and starts using it.

Here's what you need to stay ahead of this shift:

1. MCP explainer: %5Bhttps://www.youtube.com/watch?v=a9wO6GSAoGk%5D(https://www.youtube.com/watch?v=a9wO6GSAoGk)
2. AI agents guide: %5Bhttps://www.news.aakashg.com/p/ai-agents-pms%5D(https://www.news.aakashg.com/p/ai-agents-pms)
3. Context engineering guide: %5Bhttps://www.news.aakashg.com/p/context-engineering%5D(https://www.news.aakashg.com/p/context-engineering)
4. How to build AI products: %5Bhttps://www.news.aakashg.com/p/how-to-build-ai-products%5D(https://www.news.aakashg.com/p/how-to-build-ai-products)
5. Claude Code guide: %5Bhttps://www.news.aakashg.com/p/claude-code-v21-is-insane-ai-update%5D(https://www.news.aakashg.com/p/claude-

code-v21-is-insane-ai-update)

The companies that win the next 24 months are the ones building agent-accessible surface area right now. The ones that lose are still optimizing their landing page above the fold.



](https://substack.com/@aakas

219158633) 3. post by @tfadell 4. (In)Validation: The Pivotal Product Management Skill 5. Claude Code + Analytics = Vibe PMing 6. How to align product work to business goals | Corinna Stukan (CEO, Bizzy) 7. Dan Olsen & David Bland: Vibe Coding Advice for Product Teams 8.

r/ProductManagement post by u/YesterdayDreamer 9. post by @ttorres 10. r/ProductManagement post by u/Gautamagarwal75 11. r/startups comment by u/Vast_Deal_7114 12. post by @scottbelsky 13. 75% of AI PM interviews are behavioral. 14. How to use AI for your next job interview 15. post by @lennysan