AI Product Operating System

By Ashka Stephen
Engineer-turned-PM | Vercel, Microsoft | Harvard MBA, DukeCS
See product demo here and Skills repo here.

The Problem

- Product organizations are drowning in a flood of unstructured signals scattered across 15+ tools both internally (user logs, support tickets, etc) and externally (Reddit threads, X, etc).

- **The daily reality** of a PM at a growth-stage B2B company:
    - GitHub issues pile up with feature requests, bug reports, and integration demands. Each one is a potential signal, and none of them are triaged against strategy
    - Hacker News and Reddit surface competitive threats and user pain points in real-time, but no one monitors them systematically
    - Customer discovery interviews produce rich qualitative data that lives in Google Docs or Gong Calls, rarely synthesized into actionable frameworks
    - Competitive moves happen weekly, or even daily in the age of AI: pricing changes, feature launches, partnership announcements.
    - Internal data sits in dashboards that is hard to connect to product strategy

- **The cost isn't just time or iteration velocity; it's missed signals**
    - PMs spend 15+ hours/wk on information gathering and synthesis.
    - The real damage is the competitive threat that went unnoticed for 6 weeks, the customer segment that churned because discovery insights never reached the roadmap, the feature that shipped without the market context that would have changed its priority.

- **The tools** PMs use today, like Notion, Linear, Jira, Confluence, are storage tools, rather than a new age of reasoning tools. They hold information, but they don't structure it into decisions. AI summarization tools (Notion AI, Granola transcripts, ChatGPT) reduce volume but produce generic outputs that PMs don't trust for strategic decisions (and it's safe to assume, all companies/PMs are running similar data against the same model asking the same questions).

- **The gap**: No tool connects signal ingestion → structured PM reasoning → decision-ready artifacts in a single workflow. And no tool does it automatically. Yet.

North Star

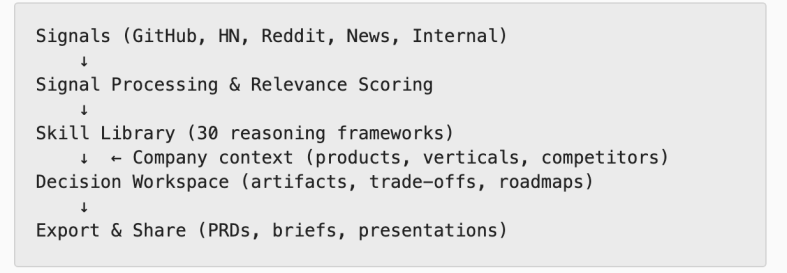
- **12-month vision**:
    - An end-to-end, self-evolving AI platform that ingests signals from everywhere a product team operates, structures them through popular PM reasoning frameworks, produces decision-ready artifacts that teams actually use, feeds those artifacts into a code-base (after human approval), submits feature changes, and eventually leads to a self-evolving product.
    - The true goal of this project would be to automate the role of what we traditionally define as a PM.
    - **North Star Metrics**:
        - **MVP: Artifact share/export rate** (show reasoning is trustworthy before anyone lets it act autonomously)
        - V2: PR Merge rate (codebase-connected)
        - V3: Autonomous actions accepted without modification (self-healing loop)
        - Final: Intervention frequency == 0

- **The key**: Value is in structured reasoning, not summarization.
    - As AI models evolve, the differentiator for AI PMs and AI products will be (1) judgment and (2) quality.
    - Both require strong context. This tool solves the context problem.
    - The value is in running that signal through a competitive response framework that classifies the threat, evaluates your moat, and produces a response strategy with trade-off documentation.

- **The approach.** A library of 30 PM reasoning skills: codified frameworks covering strategic analysis, planning, execution, agent-first product design, and research analysis. Each skill is:
    - **Principled**: Built on best practices with cited sources (prevent hallucination)
    - **Structured**: Step-by-step instructions (scored matrices, trade-off docs, roadmaps)
    - **Connected**: Skills feed into each other via a dependency graph (automation play)
    - **Contextual**: Ingest signals and company context to produce tailored outputs

This skill library *plugged into the right context* is the moat. Competitors can build signal ingestion (commodity). They can build AI summarization (commodity). They can also replicate 30 interconnected reasoning frameworks that encode how the best PMs actually think (commodity). What they can't do is plug this into specific contextual data (internal systems, customer data, usage and metric data, etc) that starts the flywheel of custom analysis.

Platform architecture:



Target Users

- **Primary persona: The Growth-Stage PM**
    - Works at a B2B SaaS company (50-500 employees)
    - Manages 1-3 product areas, reports to VP Product or CPO
    - Spends 40% of time on information gathering they wish were automated
    - Uses 8+ tools daily (Slack, Linear, Notion, GitHub, Figma, Docs, Salesforce)
    - Has tried ChatGPT/Claude for PM work but found outputs simple or generic for strategic decisions
- **Why this persona**: Large enough company to have signal overload, small enough that one PM can champion a new tool without procurement bureaucracy. The PM feels the pain personally and has authority to adopt tools.

- **Entry motion:**
    - Individual PM discovers the platform → runs their first skill (competitive-response on a real competitor)
    - Output quality surprises them (in a good way). It's structured, cited, decision-ready, not a generic summary
    - PM shares the artifact in a strategy review → team sees the quality difference
    - Team lead requests access → 3-5 PMs adopt → org-level conversation

- **Secondary persona: The VP Product** who cares about
    - consistency of PM output quality across the team,
    - reducing time-to-decision,
    - having a system of record for product strategy decisions,
    - amplifying the whole PM org.

MVP

- **Month 1: Signal Foundation**
    - **Signal ingestion**: Gather customer and product signals from public data, such as GitHub (issues, PRs, discussions), Hacker News, Reddit, etc
    - **Signal processing**: AI-powered relevance scoring, automatic tagging to skill categories, deduplication
    - **Signal feed**: Filterable dashboard showing incoming signals with source, relevance, and suggested skills

- **Month 1-2: Core Skills (5 of 30)**
    - **Customer Discovery:** interview guide generation, hypothesis tracking, synthesis
    - **Competitive Response:** threat classification, moat analysis, response strategy

    - **Feature Prioritization:** weighted scoring, RICE adaptation, trade-off documentation, sequenced roadmap
    - **North Star Metrics:** metric tree design, leading/lagging indicators, dashboard spec
    - **User Segmentation:** segment profiling, prioritization matrix, cross-segment insights
- **Why these 5?** They form a complete chain (Discovery → Segmentation → Competitive → Prioritization → Metrics) and cover the highest-frequency PM tasks.

- **Month 2-3: Decision Workspace**
    - **Artifact rendering**: Each skill produces structured artifacts (tables, scored matrices, roadmaps) rendered as interactive documents
    - **Edit & refine**: PMs can modify AI-generated artifacts, add context, adjust scores
    - **Share & export**: PDF export, Notion integration, shareable links
    - **Skill chaining**: After completing one skill, the platform suggests the next skill in the dependency graph and pre-populates context
    - **Connect to internal data systems**: Usage data, user data/logs, customer support tickets, HR systems, etc to generated an even more detailed layer of insights.
    - **Connect to internal codebase**: To create self-healing infrastructure (PM skills decide high pre features → submit PRs → collect customer feedback → iterate again)

Metric	Target	Rationale
Weekly Active Users	200	50 teams × 4 PMs — validates organic adoption
Time-to-Artifact	<15 minutes	Current manual process: 3-4 hours
Artifact Quality Rating	3.8+ / 5.0	Rated by PMs who use the output in real decisions
Skill Completion Rate	>70%	Users finish the full skill flow, not just start it
Organic Sharing Rate	>30%	Artifacts shared with at least one team member

Tradeoffs

5 skills vs. 30 skills at launch
- Choosing: 5 skills, polished
- Why: Each skill needs domain-specific prompt engineering, artifact templates, and quality validation. 5 excellent skills > 30 mediocre ones. Expand post-PMF.

Pre-built connectors vs. API-first
- Choosing: Pre-built connectors (GitHub, HN, Reddit)
- Why: PMs want zero-setup signal ingestion. API-first serves developers, not our primary persona. Add API in V2.

Single-player vs. team-first

- Choosing: Single-player with sharing to promote network effects
- Why: Adoption starts with one PM. Team features (shared workspaces, role-based access, team dashboards) add complexity that slows individual TTV. Add team features when organic sharing rate exceeds 30%.

Web app vs. CLI

- Choosing: Web app
- Why: Our target persona lives in browsers, not terminals. CLI version exists (pm-playbooks as Claude Code skills.md files) but serves developer-PMs. Web app is the mass-market path. Plus, I believe CLIs are a short-term move.

Opinionated frameworks vs. customizable

- Choosing: Opinionated with escape hatches
- Why: The value prop is structured reasoning from proven frameworks. If everything is customizable, we're just another AI text tool. Ship opinionated defaults, allow weight/criteria adjustments within the framework. Also, the strongest PMs have a POV (that is the job).

Open Qs

- **Trust calibration**: How much do PMs trust AI-generated artifacts?
    - Plan: instrument trust signals (edit rate, share rate, reuse rate) from day 1.
- **Framework rigidity vs. flexibility**: Some PMs will want to add custom scoring criteria or skip steps. How much customization before we lose the "structured reasoning" value prop?
    - Plan: start rigid, observe where users consistently override, then open those specific escape hatches.
- **Pricing model**: Freemium (limited skills) vs. trial (full access, time-limited) vs. usage-based (per artifact)? Leaning freemium; let PMs experience one full skill for free, then gate the library.
    - Plan: Need pricing-and-monetization analysis with real willingness-to-pay data.
- **Skill quality maintenance**: As the library grows to 30+ skills, how do we ensure quality stays high? Each skill needs periodic review against evolving best practices.
    - Plan: version skills, track artifact quality ratings per skill, flag skills below 3.5 threshold for review, add a meta-skills that helps skills hit a self-evolving loop.
- **Enterprise data security**: B2B PMs will feed competitive intelligence and internal strategy into the platform. SOC 2 is table stakes. Do we need on-prem deployment option for enterprise?
    - Plan: cloud-only for MVP, evaluate on-prem demand signal at 50+ paying teams.