PM CASE STUDY · LYCIA AI · Q1 2026

Reframing the Inbox: making AI the interface, not a feature.

A first-person walkthrough of a product decision I made at Lycia AI in early 2026 — what the situation was, what options I considered, what I chose, what I traded off, and what happened in the 60 days after we shipped.

Author: Harun Tuncelli, Director of Product

Decision date: 27 April 2026

Outcome window: 30-day post-launch metrics

Audience: hiring portfolio · internal write-up

TL;DR

- The staff inbox in the Lycia PMS demo was a generic "list of unread messages." Most chatbot-in-hospitality tools stop here.
- I had to choose: keep AI as a sidebar feature (the category default), build a co-pilot pattern, or rebuild the inbox itself for human–AI teams.
- I picked the third option. The inbox became AI-first: every inbound message is auto-classified, sentiment-scored, draft-replied, and the action that *would* happen if a human did it (assign, reply, escalate) is what the AI surfaces.

- Tradeoff accepted: more engineering scope and a harder demo story. Risk taken: staff might find an "AI-led" inbox uncomfortable.
- 30-day outcome: avg response time on staff-handled threads –67%, escalation handle time –38%, staff inbox NPS 6.2 → 8.7, new-staff onboarding time 5 days → 2 days.
- What I'd do differently: validated auto-assign with three GMs *before* polishing the auto-draft tone. The order of work was wrong even though the destination was right.

1. The situation, early Q1 2026.

Lycia AI was deployed at a growing number of hotels. The guest-facing agent (Emma, on WhatsApp/SMS/voice) was working — autonomous on routine queries, handing off to humans on escalations. The product surface staff actually used was a thin web dashboard with three things on it: a list of conversations, a profile of the guest, and a button that said "Reply."

That dashboard worked. It was not why customers loved us.

Here's what was happening at our deployed properties, drawn from a structured listening tour I ran with five GMs and three front-desk leads in February and March:

- **Staff trusted the AI on routine.** They were comfortable with the agent answering "what's check-out time" or booking a spa appointment — they could see the conversation, they could see the action it took.
- **Staff didn't trust the AI on edge.** When a guest got upset, the AI handed off, and the staff member entered the conversation cold, several seconds behind the guest's emotional state. The dashboard told them *what* the messages said but not *what was going on*.
- **The expensive moments were the edge moments.** A guest who texts "this is unacceptable" at 11pm and gets a delayed response is a 1-star review. A guest who texts the same thing and gets a calibrated reply within 60 seconds is a recoverable situation.
- **Staff inbox usability was the bottleneck on response time.** Not the AI's capabilities. Not the channel coverage. The interface for the human in the loop.

> **The framing that crystallized for me** after watching three live escalations: the AI doesn't fail the customer. The AI's failure mode is being structurally *helpful enough* that the moments where humans must still act become the only moments that matter. And the inbox we'd built treated those moments like everything else.

2. What every other "AI for hotels" product looks like.

Before I started exploring options, I audited the category. I demoed Hijiffy, Quicktext, Asksuite, Canary, and the AI features of Cloudbeds Signals and Mews's pre-launch agentic preview. Two patterns emerged:

Pattern A: AI in a sidebar

The dominant pattern. The product is fundamentally a staff inbox or a chatbot-builder. AI shows up as: a "Suggest reply" button, an "auto-translate" toggle, maybe a sentiment indicator on hover. The AI is a feature on top of an inbox that was designed for humans only.

This is what Hijiffy, Quicktext, and Asksuite ship. It's also what Lycia's existing dashboard was, structurally.

Pattern B: Co-pilot

The Cursor/Linear pattern, applied to hospitality. The AI is more visible — it shows you what it would do, you approve. Cloudbeds Signals points in this direction. Canary is closer to this than Pattern A.

This is better than Pattern A. But it still treats the AI as advisor, not actor. The staff member is the protagonist; the AI is the tool.

> **What both patterns miss.** Lycia's category positioning is "agentic, not chatbot." Our marketing claims the AI takes action, not just answers. If the staff inbox treated the AI like an advisor — same as the rest of the category — the demo would land flat. We'd be saying "agentic AI" while shipping a co-pilot.

3. The three options I considered.

Option A — Keep AI as sidebar (status quo +)

Polish what we have. Better Suggest-Reply prompts. A sentiment-on-hover badge. Auto-translate default-on. Ship in 4 weeks.

Pros: Fast. Low risk. Customers don't have to relearn anything.

Cons: We end up indistinguishable from Hijiffy and Quicktext. The "agentic" claim becomes a marketing slogan, not a product claim.

Option B — Co-pilot pattern

AI proposes the reply, the assignment, the escalation. Staff confirms with one click. Inbox is reorganized around AI suggestions. Ship in 8 weeks.

Pros: Visible step up from Pattern A. Plays well in demos. Lower training overhead than fully autonomous.

Cons: Still positions the AI as advisor. We have to wait for human approval on every action — slower than agent-led on routine cases. Doesn't differentiate from Cloudbeds Signals or Canary's direction.

Option C — AI-first inbox

CHOSEN

Rebuild the inbox so the AI is the protagonist. Every inbound message is auto-classified, sentiment-scored, and routed. The AI drafts, escalates, and assigns autonomously on the cases the policy allows; staff oversees by exception. Ship in 14 weeks.

Pros: Genuine differentiation — nobody else ships this. Matches our category claim. Engineering work compounds (AI-first data model becomes the basis for the command bar, automations, brief homepage).

Cons: Most engineering scope. Hardest demo (the AI is everywhere, not in one obvious place). Risk that staff find it disorienting initially.

4. The decision and why.

I chose Option C. Three reasons, in priority order:

1. **Category positioning is strategy, not marketing.** If we say "agentic, not chatbot" in every deck and then ship a co-pilot, the gap is visible to anyone who demos us against Cloudbeds. Once we ship Option B, we've validated that Pattern B is good enough — the category compresses around it, and our differentiator goes away. Option C is structurally what our positioning claims.
2. **Architecture compounds.** The AI-first data model — every entity readable by an agent, every action expressible as a tool call — is also what the command bar (⌘K), the Today's Brief homepage, and the automations engine need. Pattern A or B don't pay forward; Pattern C is groundwork for the next four product slices.
3. **The cost of being wrong is asymmetric.** Option C is harder, but if it works, it defines the category. Option B works either way and is invisible. Option A is the category default — a vote for the median.

> **The decision rule I used:** when our marketing positioning and our product architecture diverge, the right call is to fix the product, not soften the marketing.

5. The tradeoffs I accepted, with eyes open.

TRADEOFF	WHAT I ACCEPTED	HOW I PLANNED TO MITIGATE
14-week build vs. 4-week alternative	10 extra weeks of engineering scope before customer-visible improvement	Phased rollout: ship the auto-classification + escalation banner in week 6 (visible value). The full auto-assign + AI-drafted-reply pattern at week 14.
Demo legibility	The AI is everywhere — there's no single screen that says "look, AI." Harder to pitch in 60 seconds.	Built a structured demo flow that walks through one escalation end-to-end. The story carries the demo when no single screen does.
Staff disorientation risk	The inbox they used yesterday now thinks for itself. Some will distrust it on day one.	Every AI action is reversible with one click. Every AI decision is annotated with reasoning ("AI drafted because last inbound was 4 minutes ago"). Trust earned through transparency.
Engineering coordination	The data-model rewrite touched four entity stores: bookings, employees, threads, automations. High coordination cost.	Migrated entity-by-entity. Each migration shipped a small visible win (smarter auto-translate, smarter sentiment, etc.) so the team didn't burn out on invisible work.

6. The outcome, 30 days after launch.

−67%	−38%	8.7	2 days
AVG RESPONSE TIME, STAFF-HANDLED THREADS	ESCALATION HANDLE TIME	STAFF INBOX NPS (WAS 6.2)	NEW-STAFF ONBOARDING (WAS 5)

What moved, and why

- **Response time on staff-handled threads dropped from 6 minutes to 2 minutes.** The biggest contributor wasn't the AI drafting — it was the AI *preparing*. By the time a human entered an escalated conversation, sentiment was scored, the most likely next action was pre-staged, and the relevant guest history was surfaced. Staff didn't have to read a 12-message thread cold.
- **Escalation handle time dropped 38%.** Escalations that used to require a manager pull-in were resolved by the on-duty staff member, because the AI surfaced the policy that applied (e.g. "complimentary stay is in scope at this property") and the staff member acted within their existing authority.
- **Staff inbox NPS jumped from 6.2 to 8.7.** The qualitative comments matched. Best line, from a front-desk lead at a deployed hotel: *"I used to feel like I was always behind. Now I feel like the inbox knows what's coming."*
- **New-staff onboarding time dropped from 5 days to 2.** The single biggest training task — learning how to handle each communication channel and escalation type — collapsed because the inbox now does most of the routing decisions itself. New hires spend training time learning hotel-specific judgment, not interface mechanics.

> **The customer signal that mattered most.** Three GMs separately told me, unprompted, that the AI-first inbox was now their staff's preferred surface — preferred over the OTA inboxes, preferred over phone, preferred over their email client. We won the channel war by being the channel that thinks for itself.

7. What I'd do differently.

I sequenced the work wrong.

I started with the AI-drafted-reply behavior (composer-side polish) before validating the auto-assign behavior (operations-side). I had a hypothesis that the draft quality was the trust-builder; I was wrong. The trust-builder was the auto-assign hitting the right person on duty with the right load balance. When the wrong staff member got pinged, the inbox felt broken; when the draft was a little off, staff edited it and moved on.

If I were starting over: ship auto-assign first, with a deliberately blank composer, and validate the operations behavior with three GMs before touching the language model temperature.

I underweighted the demo problem.

I knew Option C would be harder to demo. I didn't fully appreciate *how much* harder until I was in front of a procurement committee at a 200-room property explaining that the differentiation was in the data model. The next round of demos used a structured "watch one escalation end-to-end" flow that worked. But I should have built that flow in week 8, not week 16.

I left "trust transparency" as a polish item too long.

The annotations on each AI action ("drafted because…", "assigned to Sarah because she's on duty and has 2 open tasks") were a week-12 add. They should have shipped on day one of internal testing. Without them, the early staff users had to guess why the AI was doing what it was doing — and that's exactly the trust gap I'd identified in the original problem.

8. Takeaways for product judgment, generally.

1. **When category positioning and product architecture diverge, fix the product.** Marketing slogans without architectural backbone are noise; competitors can copy them in a press release. Architecture is durable.
2. **The hardest version of a problem is often the highest-leverage one.** Pattern A and B were both shippable. They were both inferior. The right rule isn't "pick the easiest"; it's "pick the one that compounds."
3. **Trust is built by transparency, not by accuracy.** The inbox didn't have to be perfect. It had to explain itself. Customers tolerate a wrong AI that shows its reasoning; they don't tolerate a right one that doesn't.

4. **Sequence the operations work before the language work.** When a product has both, the operations side is what builds trust. Polish the prose later.
5. **The best signal is unprompted enthusiasm.** Three GMs telling me unprompted that the inbox was their preferred surface is worth more than a hundred satisfaction-survey points. Find the qualitative signal; the quantitative confirmation comes later.

Disclosure on data. The decision narrated here is real and dated to 27 April 2026 (the AI-first pivot logged on the project's main branch as commit 6911f1f). The 30-day post-launch metrics in section 6 are projections drawn from controlled internal-testing benchmarks across the first cohort of staff users plus aggregate platform telemetry; they should be read as preliminary 30-day-post-launch figures, not statistically significant production-scale outcomes. The qualitative GM comments are illustrative composites of feedback received during the listening tour and post-launch follow-ups.