KELENO

INSIGHTS · AGENTIC AI

The Agentic AI Playbook.

From pilots to production in Hospitality & Healthcare — a practical guide to deploying AI agents inside real operational systems, without breaking what already works.

PREPARED BY
Keleno — AI & Agentic Engineering

FOR
Operators & technology leaders

READING TIME
~12 minutes

What's inside.

A short, practical walk from what "agentic" actually means, through two real deployments, to a checklist you can run against your own roadmap this quarter.

Agent, not chatbot.

Most "AI" a guest, patient, or front-desk team has touched so far is a chatbot: it answers a question and stops. An agent is built to finish the task.

A chatbot retrieves information and hands the decision back to a human. An **agent** is given a goal, a set of tools, and the authority to act — it reads a request, decides which system to touch, takes the action, and confirms the outcome, looping in a person only when the situation calls for it.

That distinction is the whole ballgame operationally. A chatbot lives safely in a sandbox. An agent has to be wired into the systems that actually run the business — a property management system, a booking engine, a patient data platform — which means it inherits all of that system's stakes: security, auditability, and the cost of getting it wrong.

This is also why "agentic AI" gets harder to ship the further it moves from a demo. Answering a question in a chat window is low-risk. Changing a hotel room assignment, or deciding how a lab result gets routed between two healthcare systems, is not — and it shouldn't be treated the same way.

→ Chatbot

Answers questions. Stateless per session. No write access to operational systems. Human does the work after.

→ Agent

Pursues a goal. Holds context across steps. Has scoped write access to real systems. Human reviews by exception.

Why most pilots never leave the sandbox.

The gap almost never shows up as a model problem. It shows up as an integration and governance problem, discovered too late.

Enterprise AI pilots are easy to make look good. Point a model at a curated dataset, script a happy-path demo, and it will impress a room. Production is a different test entirely: real data with real gaps, systems that were never designed to be called by a machine, and a business that can't tolerate the agent guessing wrong on a live booking or a patient record.

In our experience, four things quietly kill more pilots than the AI itself ever does:

01 NO INTEGRATION PATH

02 NO AUDIT TRAIL

03 UNSCOPED AUTHORITY

04 NO OWNER POST-LAUNCH

The pilot was built against a mocked-up version of the PMS or the patient record system, not the real, messy, rate-limited one. Nobody can later explain why the agent made a specific decision — a blocker the moment compliance or a GM asks. It was given broad access "to keep things simple," which is precisely what makes security review reject it. And once the demo lands, there's no team responsible for watching what it does in the wild.

None of these are AI problems. They're systems-engineering problems — which is exactly why they're solvable with the right delivery discipline, not a bigger model.

A guest-facing agent, inside a live PMS.

DELIVERED · VIVIDITY, BUILT FOR HUBLOFT

Hubloft needed more than a booking chatbot — guests wanted to change a room, request a late checkout, or ask for housekeeping in the same conversation they used to ask about parking. Answering the question wasn't the hard part. Acting on it, inside Hubloft's connected property management systems, was.

We built **Vividity**, a guest-facing conversational agent that hooks directly into multiple PMS platforms. When a guest asks for a room change or a service request, Vividity doesn't just draft a reply — it takes the action in the connected system, within the permissions it's been scoped to, and confirms back to the guest in the same thread.

The engineering problem underneath the friendly conversation: normalizing very different PMS APIs behind one consistent tool interface, handling partial failures gracefully (a system is down, a room type is sold out), and giving hotel staff visibility into every action the agent took on their behalf.

> *"Working with Keleno felt like having a world-class product engineering team embedded within our own company."*
>
> CEO & CO-FOUNDER, HUBLOFT

An agent that decides how data moves.

DELIVERED · MEDSYNCAI

MedSyncAI, a California-based healthcare software provider, needed a data exchange layer that could sit between very different healthcare systems — each with its own ingestion format and its own rules for how information should be distributed downstream. Hardcoding that logic per integration doesn't scale, and healthcare data can't tolerate a system that guesses.

We built an agent whose job is narrower and more disciplined than a general assistant: given an incoming record, it decides the correct ingestion method and the correct downstream routing and distribution — autonomously, but inside rules that are explicit, logged, and reviewable. Users configure how outcomes should be distributed; the agent's role is to apply that configuration correctly and consistently, every time.

This is agentic AI at its most conservative and, we'd argue, its most valuable: not a system that talks to patients, but one that removes a specific, error-prone, manual decision from a compliance-sensitive workflow — with a full audit trail behind every choice it makes.

> *"Keleno demonstrated precision, security, and speed from day one."*
>
> CEO & CO-FOUNDER, MEDSYNCAI

Five steps from pilot to production.

The same sequence held for both a guest-facing agent and a back-office data-routing agent. It's the sequence, not the use case, that determines whether an agent ships.

1 Scope the task, not the department

Pick one decision or one action the agent owns end-to-end — a room change, a routing decision — instead of "customer service" or "data ops" broadly.

2 Integrate against the real system first

Build and test against the actual PMS, EHR, or data platform — rate limits, auth quirks, and all — not a mock. This is where most timelines are actually spent.

3 Scope its authority explicitly

Define exactly which actions it can take unattended, and which require a human sign-off, before the first real request arrives — not after an incident.

4 Log every decision, not just every answer

An agent needs the same audit trail a careful employee would leave — what it decided, why, and what it touched — reviewable by someone who wasn't in the room.

5 Assign an owner for week two

Launch day is easy to staff. Decide who watches the agent's decisions a month in, and what triggers a re-scope — before it's needed.

Where to automate — and where not to.

Agentic AI isn't a maturity ladder you climb to the top of. Some decisions should stay fully automated; others should never leave a human's hands. Knowing which is which is most of the job.

Good fit for autonomy

Rule-governed routing decisions · rebooking within defined inventory · status updates and confirmations · data ingestion & format normalization · repetitive requests with a clear, bounded action space.

Keep a human in the loop

Anything touching a refund or a financial exception · clinical judgment or diagnosis-adjacent decisions · first-time or ambiguous requests · anything where being wrong is expensive to reverse.

In regulated environments especially — healthcare above all — the right default is narrow, well-logged autonomy over broad, general-purpose autonomy. An agent that reliably does one thing correctly, with a clean audit trail, earns the right to take on the next thing. An agent given broad discretion on day one rarely survives its first security review.

Are you ready to move past the pilot?.

If you can check most of these, your pilot is closer to production than it might feel.

- [ ] **Scoped task:** the agent owns one clearly defined action or decision, not a whole department.
- [ ] **Real integration:** it's been tested against your live PMS / EHR / platform — not a mocked version.
- [ ] **Explicit authority:** what it can do unattended, and what needs sign-off, is written down.
- [ ] **Audit trail:** every action it takes is logged in a way someone outside the project can review.
- [ ] **Failure handling:** you know what it does when the downstream system is slow, down, or returns something unexpected.
- [ ] **Named owner:** someone is accountable for how it performs a month after launch, not just on launch day.
- [ ] **Compliance sign-off:** security and (where relevant) compliance have reviewed it — before go-live, not after.

Missing two or three of these isn't a red flag — it's a punch list. It's usually a matter of weeks, not a re-architecture, to close the gap.

KELENO

LET'S TALK

Have a pilot that needs to earn production access?

Tell us what you're running today — the PMS, the patient data platform, the workflow. We'll tell you honestly whether an agent belongs in it yet, and what it would take to do it right.

Talk to an engineer — keleno.com/contact.html

Email
info@keleno.com

Phone / WhatsApp
+91 88483 58534

Delivery HQ
Kochi, Kerala, India

Client Ops
West Lafayette, IN, USA