Most AI Pilots Stall in Development. Here's How One Made It to Production.

How an agentic AI architecture, rigorous evaluation discipline, and production-grade engineering shipped a public-facing healthcare data chatbot for a state government and what it means for enterprise AI teams everywhere.

Author: **Mitch Quinn, Director of AI/ML**

Somewhere between the executive demo and the production launch, most enterprise AI projects quietly stall. The capability works. The proof-of-concept impresses. But **the gap between we built something that works in the demo** to the executive committee **and we deployed something into production** real users will rely on tomorrow **turns out to be wider than anyone budgeted for.**

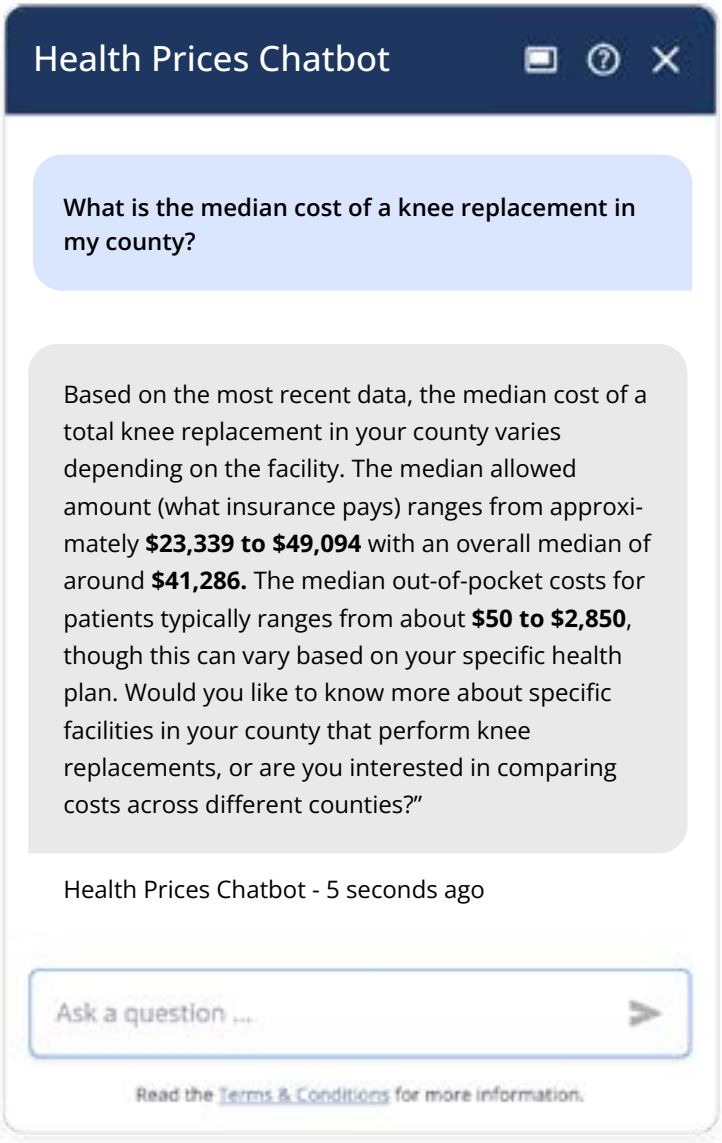
We were brought into a project that was sitting squarely in that gap.

A data analytics firm had committed to building a public-facing AI assistant for a state government: a chatbot that would let any resident, legislator, or policy researcher ask plain-language questions about healthcare cost and quality data and get a real answer back based on data collected about payer claims, pricing, utilization and provider quality for that state. They had a working prototype. They had momentum. What they needed was a partner who had taken AI systems like this across the production line before and who could help them think through the parts that don't show up in the demo: evaluation, guardrails, scale, change management, the long tail of what happens when this is live. That partnership is the case study.

Picking the right architecture for the data

Their early prototype used a retrieval-augmented generation (RAG) approach: load the source data into a vector store, retrieve relevant chunks at query time, let the language model synthesize an answer. It's a sensible default, and it's the right answer for a lot of problems.

But wasn't the right answer for this problem.

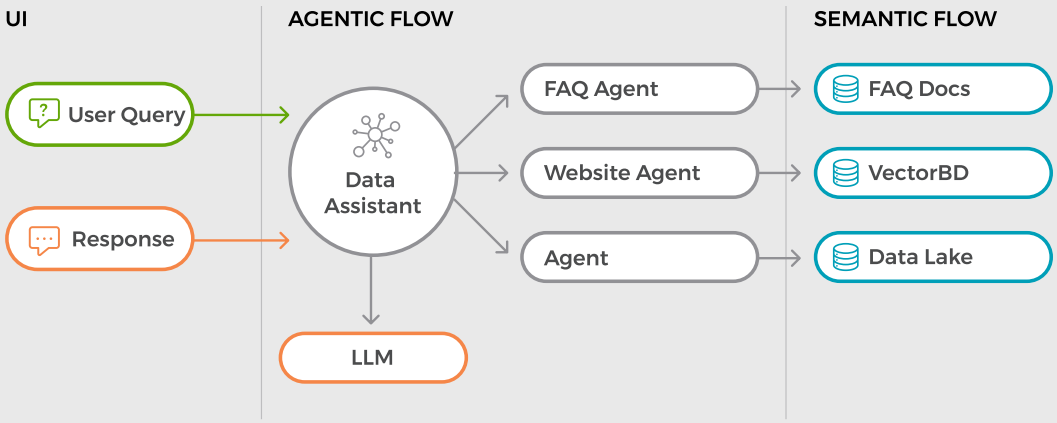


Chatbot & Agentic Data Assistant

TYPES OF QUESTIONS

Can I get a list of primary care physicians in this area code?

What is the typical cost of a knee replacement in this state?



The soution is a web-based chatbot interface designed to answer questions from patients, policymakers, and researchers alike. Its architecture is built for flexibility, allowing new datasets to be integrated over time as needs evolve. To ensure reliability and trustworthiness, the system incorporates a rigorous evaluation methodology, framework, and guardrails throughout.

RAG works well when your data is genuinely unstructured (documents, policies, articles, the kind of content where semantic similarity is what you want). **But healthcare cost and quality data isn't unstructured. It's tabular, relational, and exact.**

When a user asks "what are the five most expensive facilities for knee replacement in this region," there is a correct answer, and it lives in a database, and the way to get it is a SQL query.

Together with the client, we redesigned the chatbot as an agentic system. The agent reads the user's question, decides what kind of question it is, writes an actual query against the actual database, runs it, and summarizes the result. For documentation lookups (methodology questions, FAQs, the meaning of acronyms), a separate sub-agent handles that path. A web crawler tool handles questions that require reaching beyond the data into related state resources. An orchestrator at the front routes traffic between them.

We proved it out in days. A quick proof of concept using different foundation models for comparison, run against the same questions the team had been wrestling with, returning the right answers cleanly. The architectural shift unlocked everything that came next.

The part nobody talks about

Here is what kills most enterprise AI pilots: there is no rigorous way to measure if the system is getting better or teams underestimate the rigor that is needed to deploy into production.

Teams keep falling into the same trap. They run a few queries, eyeball the responses, declare it "feeling better". Then they tweak a prompt and it feels better again. Then they swap a model and it feels better still. Six weeks in, no one can tell you whether the system is more accurate than it was at the start, or whether the last three "improvements" silently broke something that used to work.

You can't improve what you can't measure, and you can't measure conversational AI the way you measure traditional software. There's no clean pass/fail. There's no exact match. You need a different kind of test, built deliberately, run constantly.

So we built the evaluation harness alongside the system itself. The team assembled roughly a hundred question-and-answer pairs covering the data the chatbot would actually be asked about: questions from real users in usability testing, questions from the legal review, questions from people trying to break it. Each one had a known-good answer. Every code change, every prompt change, every model swap ran against the full set, with a separate language model acting as judge: did this response match what we expected, yes, no, partially?

A few things this gave us that mattered:

The first was directional truth. When we tested a faster, cheaper model, an open-weight one with much better latency, the eval told us immediately that overall correctness had dropped. Not "it feels a bit off." A number, lower than yesterday's number, broken down by question category so we could see exactly which kinds of questions had gotten worse.

The second was a category breakdown that drove the roadmap. The eval showed us that questions answered by querying the structured data were landing reliably above 85% accuracy. Questions that depended on less structured methodology lookups were closer to 60%. That told us where to spend the next month of engineering, not based on intuition but on data.

The third was a guardrail suite. A separate set of evaluations dedicated entirely to questions the chatbot must never answer. Lottery numbers. Medical advice. Anything outside the scope of public healthcare data. Every change ran against the guardrail set too. We could prove, on demand, that the system declined what it was supposed to decline.

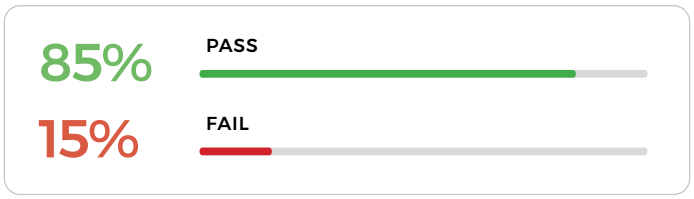
When you're deploying a public-facing AI product on behalf of a state agency, the question stops being "is it good?" and becomes "can you prove it's good, can you prove it's still good after this change, and can you prove it isn't doing the things it shouldn't?" Without an evaluation discipline, those questions don't have answers. They have handwaving.

Architecture decisions that age well

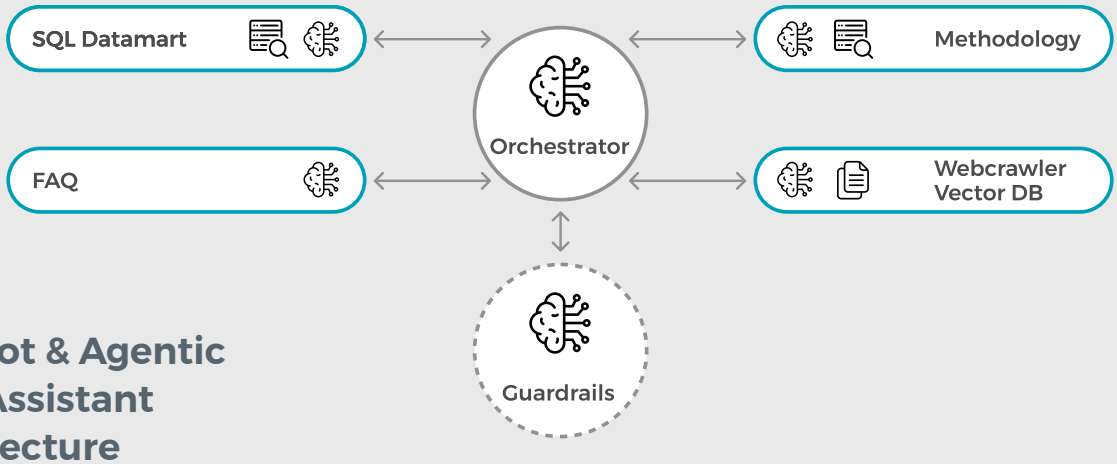
A few quieter choices turned out to matter more than expected. **We enforced security boundaries in code, not in prompts.** Modern language models can be talked into a great many things, and "the system prompt told you not to" is a brittle defense against a determined user. Data access is filtered at the database layer (what columns, what tables, what rows) based on credentials passed through the API, not on instructions in the prompt. When the system later moved into a more sensitive internal product line with role-based access, this design choice was the difference between a refactor and a rebuild.

We constrained the SQL the agent could generate. Single SELECT statements only. No inserts, no updates, no schema changes. The agent's query is parsed and validated before it ever reaches the database. If the model hallucinates something destructive, the destructive thing never runs.

We separated the response format from the interface. The chatbot returns markdown. The front end renders it however it likes. This sounds trivial; it isn't. It meant when the client decided they wanted the same backend to power a different UI on a different product, the agent didn't have to change. The contract was the contract.



Trace ID	Request	Response	Exec Time	State	Correctness
AC-da765f	Which payer has the largest average gap betw ...	Based on the data, Alpha Health has the ...	7.793s	OK	Pass
AC-brMG5r	Which payer has the highest out-of-pocket m ...	Based on the data, Alpha Health has the ...	8.929s	OK	Pass
AC-Sd5462	Which payer has the highest hospital claim vol ...	Based on the data, Alpha Health has the ...	8.604s	OK	Pass
AC-Mh5f21	Which payer and health plan combination has ...	Based on the data, Alpha Health has the ...	8.13s	OK	Pass
AC-d8387g	Which hospital that performs colonoscopies ...	Based on the data, there are several hospitals ...	9.261s	OK	Pass
AC-838i5fg	Which hospital service has the widest average ...	Based on the data, spinal fusion has the w ...	8.307s	OK	Pass
AC-br10fg	Which hospital service has the most distinct p ...	Based on the data, -Screening Digital Mamm ...	6.764s	OK	Pass



Chatbot & Agentic Data Assistant Architecture

We kept the system prompt out of the codebase. Late in the project, we decoupled prompt management entirely so the team could iterate on instructions without redeploying. This is one of those decisions that costs you a day in week three and saves you a quarter in month nine.

None of these are exciting. All of them are the reason the system is still running.

What happened after deployment

The chatbot has been live on the state's public healthcare data portal for about a month. The first few weeks were quiet: a dozen messages a day, mostly people who'd stumbled across it. Then the agency formally announced it, and traffic climbed sharply. It has since settled into a steady rhythm of real residents asking real questions and getting real answers.

The work was recognized with a national industry award. Another state has approached the client to build something similar.

Inside the original engagement, the project earned an expansion. The same client is exploring whether this same architecture could replace their longstanding analytics dashboard product entirely: a multi-year shift from licensed BI tools to an AI-native data exploration experience. That conversation only happened because the first thing deployed, and deployed well.

What this means if you're leading AI in your organization

Most of what's published about enterprise AI right now is about capability: what the models can do, what the new release supports, what's possible. That's not the bottleneck. The bottleneck is the gap between a capability and a product your name goes on. Between a notebook and a system. Between a demo that wows the executive team and something that holds up when real users start using it.

Closing that gap is not primarily an AI problem. It's an engineering discipline problem in new clothes. Evaluation harnesses are regression tests. Guardrail suites are security tests. Decoupled prompts are configuration management. Programmatic access controls are the same controls you've enforced on every system you've ever deployed to production. The technology is new. The discipline is the same one that has always separated production software from prototypes.

The teams that get AI to production are not the teams with the best models. They're teams who architect, build and test AI products with the same rigor, safety and security standards as traditional software products, but with modified tools and techniques that apply specifically to applied AI products.

Mitch Quinn
Director of AI/ML

Let's get to work.

Every successful partnership starts with an introduction. Send us a message or schedule time to chat.
contact@xby2.com