EXECUTIVE BRIEFING

KOHL DIGITAL

AI Knowledge Systems for Customer Support

A Builder's Guide

How to get AI to answer customer questions without wrong answers: the four parts, a test gate, a humans-first launch and the four numbers to track.

Mike Kohl
Founder, Kohl Digital

mikekohl.com/guides/ai-knowledge-systems-for-customer-support

KEY TAKEAWAYS

- Build four parts with named owners: approved sources, governance rules, an answer layer that uses only those sources, and escalation paths to people.
- Pass a test suite of real questions with known answers before launch; the reference build passed 370 questions with zero errors.
- Launch with humans reviewing every answer, lighten review per question type only on evidence, and track share answered, error rate, speed and escalation.

To get AI to answer customer questions without wrong answers, build a governed knowledge system. It has approved sources with named owners, an answer layer that draws only on those sources, a test suite of real questions with known answers that must pass before launch, and escalation paths for every question the system should hand to a person. Humans review every answer at launch, and autonomy grows from evidence.



A governed system has four parts, each with an owner, behind a test gate.

If you are a COO or head of customer support, you own a number: the share of customer questions that reach your staff. This guide shows how to lower that number and keep the answers right. It is the overview for a cluster of six deeper guides, linked at the end.

What a governed knowledge system is

A governed knowledge system is a company's answers, organized so software can use them safely. Every fact in it has a source, an owner and a date. History is kept apart from current truth, so an old policy never answers a new question. Access follows your rules, so a customer sees only what a customer should see.

Governance is the word for the habits that keep those properties true after launch. Someone owns each answer. Someone reviews changes. Someone can say why the system gave a specific answer last Tuesday.

A chatbot pointed at a shared drive has none of that. It will answer fluently from whatever it finds, including the policy that was replaced two years ago. Fluency makes the wrong answers harder to catch.

The four parts

A governed system has four parts. Each one has a job and a person responsible for it.

Part	Job	Who owns it
Sources	The approved answers, policies and product facts, each with an owner and a date	Content owners in support, product and policy
Governance	Rules for review, change, conflict and access	Head of support or operations
Answer layer	Reads the question, finds the right approved material and drafts the reply	Engineering lead
Escalation	Routes questions the system should not answer to the right human	Head of support

Sources

Sources are the raw material: resolved tickets, help articles, policy documents, price sheets. Pick a short list of sources you will stand behind and retire the rest. A smaller set of trusted material beats a large pile of mixed quality.

Governance

Governance answers four questions for every fact: who wrote it, who owns it, when was it last confirmed, and what replaces it when it changes. The guide on keeping company knowledge current turns these into a weekly routine.

Answer layer

The answer layer includes an intent classifier, which reads each incoming question and decides what kind of question it is. The kind decides what happens next: answer from approved material, ask a clarifying question, or

escalate. Classification is what lets the system treat a refund question differently from a how-to question.

Escalation

Escalation is designed before launch. A question the system declines needs somewhere to go, and the person who receives it needs the context. The guide on human review and escalation covers the three layers in detail.

Start from the questions your customers already asked

Your history already holds the questions that matter. Start there, so the system is built for the real mix of questions you receive and none you imagined.

On a build for a client, we started from 7,519 historical customer question-and-answer records. Those records told us what customers ask, in their own words, and how staff answered. They also showed which questions repeat and which are rare.

Here is the procedure a team can run on Monday.

1. Export a year of resolved questions and the answers staff gave. Include the channel and the date.
2. Group the questions by intent. Count how many questions each intent holds, using your own data.
3. For each intent, find the approved source of truth. Write down the owner and the date it was last confirmed.

4. Mark each intent as answer, answer with review, or escalate. Anything involving money, safety, legal exposure or a changed policy starts as escalate.
5. Flag every question where staff gave different answers over time. Those are conflicts to resolve before the system sees them.

Step 5 pays for itself early. Conflicting past answers show where your policy was unclear, and a system trained on the conflict will reproduce it.

The test gate

Before launch, the system must pass a test suite of real questions with known answers. Ours was 370 questions drawn from real customer history, each with a verified answer. The system passed with zero errors. Launch waited until it did.

A test gate turns "it seems to work" into evidence. It also gives every later change a pass or fail check, which matters because every policy update, new source and model change can break something that worked yesterday.

The suite should include the hard cases: questions where the policy changed, questions with ambiguous names, questions where two sources disagree, and questions the system must refuse. The guide on how to test an AI knowledge system before launch lays out the method step by step.

Launch with humans first

At launch, a human reviews every answer before or immediately after the customer sees it. This feels slow, and it is the fastest way to find the failures your test suite missed. In our build, humans reviewed every answer at launch.

Autonomy is then earned per question type. When a type of question has a long run of reviewed answers with no corrections, it moves to lighter review. A type that produces corrections stays under review, and each correction becomes approved guidance the system reads next time.

The result in that build: the system now answers more than half of the customer questions that used to reach staff. The remaining questions still reach people, with context attached, and your best staff spend their time on them.

Who does what

A knowledge system touches several teams, and it works when each role is named before the build starts.

- **COO or head of support.** Owns the share of questions answered and the budget. Chairs the weekly review.
- **Source owners.** Each owns a set of facts and confirms them on a schedule. They come from support, product, finance and policy.
- **Engineering lead.** Owns the answer layer, the classifier, the test run and the audit trail.
- **Reviewers.** Support staff who check answers at launch and record the cause of each correction.

- **Coordinator.** One person who keeps the weekly routine running and chases open items.

If any role is empty on launch day, fill it first. An empty owner role becomes a stale answer within months.

A 30-day path

The 30-Day Leverage Pilot runs this method on one workflow. Week one finds the lever and picks the intents. Week two maps the constraint and the knowledge: sources, owners, conflicts. Week three builds the system and the test suite. Week four goes live with humans reviewing every answer. At day 30 you decide to continue or stop, with measured numbers in hand.

What to measure

Four numbers tell you whether the system works. Define each before launch and record a baseline from your history.

Measure	What it tells you	How to count it
Share of questions answered	How much volume no longer reaches staff	Answered by the system divided by all incoming questions
Error rate on the test suite	Whether quality holds after every change	Failed test questions divided by total test questions, on each run
Time to answer	What the customer feels	Minutes from question to correct answer, by intent
Escalation rate	Whether the system declines at the right moments	Escalated questions divided by all incoming questions, by intent

Read them together. A rising share answered with a rising error rate means the system is overreaching. A falling escalation rate with stable errors means the system is earning trust. The guide on cost and return turns these numbers into a financial case.

What goes wrong

- **The sources disagree.** Two documents give different refund windows, and the system picks one at random. Fix the conflict at the source before launch.
- **The test suite is too easy.** A suite of friendly questions passes while real customers find the gaps. Build it from history and include refusals.
- **Nobody owns an answer.** A policy changes, the page stays stale and the system keeps quoting the old version. Every fact needs an owner with a review date.

- **Escalations arrive without context.** The specialist asks the customer to repeat everything. Send the question, the draft answer, the sources checked and the reason for escalating.
- **Review ends too early.** A team declares victory after a week and removes humans from the loop. Move a question type to lighter review only with evidence from that type.
- **Access leaks.** The system answers a customer from an internal document. Access rules belong in the system, applied to every answer.
- **Launch before the owners exist.** Sources go live with no named person behind them. Assign owners and dates before the first customer sees an answer.
- **Success measured by volume alone.** A high share answered looks good until the error rate is checked. Read the four numbers together.

Questions leaders ask

How long does this take to build?

It depends on the state of your sources and the number of integrations. The 30-Day Leverage Pilot is built to put a first version live with humans reviewing in four weeks, then decide with evidence whether to continue. A full rollout across every question type takes longer, because autonomy is earned type by type.

Do we need to clean up all our documents first?

You need to clean up the sources for the first intents. Start with the intents that cover the most volume and clean the sources for those. The rest of the library

can wait until the first intents are live and measured.

What happens when a policy changes?

The owner updates the source, the date changes, the old version moves to history, and the test suite is rerun. Any test question affected by the change gets a new known answer. The guide on keeping knowledge current covers the weekly routine.

Will the system ever say "I don't know"?

It should. Declining and routing to a person is a designed behavior, and it is the one that protects your customers when the sources do not cover a question.

The six guides in this cluster

- How to test an AI knowledge system before launch
- RAG vs. a governed knowledge system
- Build vs. buy for an AI knowledge base
- What an AI knowledge system costs and how to measure the return
- Human review and escalation design for AI answers
- Keeping company knowledge current

The Leverage Diagnostic takes about six minutes, scores the six places leverage hides and prices the gap with the math shown, which tells you whether customer support is where to start.

NEXT STEP

Find where leverage hides in your company.

The Leverage Diagnostic takes about six minutes, scores the six places leverage hides and prices the gap with the math shown.

mikekohl.com/diagnostic

Kohl Digital builds AI systems for mid-market and enterprise companies. Founder Mike Kohl architected the enterprise sales platform at Level 3 Communications and Lumen Technologies that went from $0 to $500 million in four years.

Book a call: mikekohl.com/contact