AN EDUCATIONAL BRIEF · V1.0 · FREE

Your AI Doesn't Know Your Business

A plain-English guide to RAG — the memory layer that turns a generic AI into an agent that actually knows your operation.

> *A general-purpose AI knows almost everything except the one thing that matters: your business. RAG is the memory layer that fixes that — turning a brilliant amnesiac into an agent that actually knows you.*

BY	Alex Jahn
EDITION	v1.0 — April 2026
AUDIENCE	Small business operators
READING TIME	~15 minutes
COST	Free

Why ChatGPT alone is useless for your business

You've probably tried it already. You ask ChatGPT (or Claude, or Gemini) a real business question — "what should I quote for a 1,200-square-foot kitchen remodel?" — and the answer comes back smooth, confident, and wrong. The number is plausible. It's also not *your* number. Not your market, not your suppliers, not your margins, not your reputation in the room when you hand it to a customer.

That feels like an AI failure. It isn't. It's a **missing memory** failure.

The new hire analogy

Imagine you hire a brilliant person. Twenty years of construction experience, sharp instincts, polished communication — the dream candidate. Day one, you put them in front of a customer and ask them to quote a job. What happens?

They give a confident, articulate, completely useless answer. Because they don't know your suppliers. They don't know your margins. They don't know the customer just had you redo their bathroom last spring. They don't know your foreman has a rule about not bidding window swaps in winter. **The brilliance is real and the answer is wrong** — because they have no orientation in *your* business yet.

Generic AI is exactly that brilliant new hire, every single day, forever. Smart but ignorant of your specific operation. The solution isn't to hire a different AI. The solution is to give the AI an orientation — a memory of how your business actually works.

That memory layer has a name. It's called **RAG**. The next 10 pages explain what it is, what to put in it, what to keep out, and what your agent stack looks like with one versus without one.

> **HI, I'M ALEX.** I'm a working carpenter and house-flipper in Fond du Lac, Wisconsin. I run a multi-agent operations stack for my own trade business — proposals, scheduling, jobsite notes, comp lookups, customer follow-ups. RAG is what makes any of that actually useful for my jobs instead of generically impressive. I run Agent Logic, a small consultancy where I help other tradespeople and small business operators get to the same setup. This brief is what I wish someone had handed me on day one.

> **ONE SENTENCE:**
> Generic AI is a brilliant new hire on day one, every day — and the gap between "impressive demo" and "useful for your business" is closed by giving the AI a memory of your business. That's what RAG is.

What RAG actually is

RAG stands for Retrieval-Augmented Generation. The acronym is forgettable; the idea behind it isn't.

The open-book exam

The clearest way to think about it: RAG turns a closed-book exam into an open-book exam. Without RAG, your AI is forced to answer every question using only what it learned during its original training (which ended months or years ago and never included your business in the first place). With RAG, the AI is allowed to *open the book* — pull the most relevant pages from your business's actual records — before answering.

An open-book exam doesn't make a stupid student smart. It makes a smart student suddenly capable of citing your company's actual SOPs, real customer history, and real pricing data instead of guessing.

What happens behind the scenes (in plain words)

Every time you ask a RAG-enabled agent a question, four things happen:

1. **You ask.** The agent receives your question, exactly as you typed or spoke it.
2. **The system searches your knowledge base.** Not your whole knowledge base — just the small handful of documents that are most relevant to what you asked. Think of it as "the smartest librarian in the world, in 50 milliseconds, with infinite memory of where every relevant fact lives."
3. **Those relevant pieces get added to the question as context.** Behind the scenes, the question the AI actually sees is something like: "*Here's the user's question. Here are the three most relevant snippets from their proposals folder, their materials catalog, and their customer notes. Now answer.*"
4. **The AI answers using both kinds of knowledge.** Its general intelligence (how to write an estimate, how to phrase a follow-up, how to do math) plus your specific information (your prices, your customer, your past work). The answer is grounded in your reality, not the model's guess.

Why "augmented" is the right word

The AI's general capabilities are still there — you don't lose its ability to reason, write, or summarize. RAG just *augments* the question with the right facts before the AI answers. The AI gets smarter about your business without being retrained on your business. That distinction matters for cost. We'll come back to it on page 9.

> **The mental model that sticks:** RAG = AI + an open book of your business. The AI provides the brains; the book provides the facts. Without the book, brains alone make confident-sounding guesses. With the book, brains plus facts give correct answers grounded in *your* reality.

Your business already has the knowledge

The first thing most operators say when I explain this is: "But I don't have a knowledge base. I'm not some big enterprise with a wiki." That reaction is correct in the literal sense and wrong in the practical sense. **You absolutely have a knowledge base.** It's just scattered across formats and locations none of which talk to each other.

Where your business knowledge actually lives today

Where it lives	What's in it
QuickBooks notes	Customer payment history, "always pays late," "prefers Tuesday calls," past invoice totals.
That spreadsheet	Your pricing rules of thumb. Square-foot cost estimates by trade. Your buy-box criteria. Margins by job type.
Email folders	Past quotes, customer back-and-forth, vendor correspondence, complaints, testimonials.
Phone photos	Jobsite progress, finished work, problems-found, before-and-afters, business cards from suppliers.
Your foreman's brain	"Don't bid window jobs in February." "The Schulzes always tip in cash." "Use the painter from Oshkosh, not the cheap one in town."
The proposals folder	Every quote you've ever sent, with the actual numbers, in PDFs nobody reads twice.
Your phone's notes app	Reminders, half-finished thoughts, names of subcontractors, gut-feel pricing for that one weird job last year.

The cost of leaving it scattered

- **Training a new hire takes 90+ days** instead of 30, because they're learning the business in pieces from whoever's around.
- **Your foreman quits and five years of pricing intuition leaves with him** — because none of it was written down anywhere a successor could find.
- **You re-research the same question three times a year** — because last year's answer is buried in an email thread you can't find.
- **You quote inconsistently across customers** because there's no canonical source of truth on your own pricing.
- **Your AI agent makes things up** — because none of this knowledge is reachable to it, even though it all exists.

The reframe

You don't need to *create* a business knowledge base. You already have one. What RAG does is take the knowledge you already have, sitting in scattered formats, and make it *available* — to your AI agent, to new hires, to a successor running the business when you take a vacation. The hard part isn't generating the knowledge. The hard part is making it reachable.

> **REALITY CHECK:** if you've been running your business for more than 2 years, you have between 500 and 5,000 documents of latent knowledge already created — you just can't search through it the way an AI can. RAG closes that gap.

What belongs in RAG (and what doesn't)

Just because RAG *can* hold information doesn't mean every kind of information belongs there. The fastest way to ruin a RAG system is to dump everything in and hope. The second-fastest way is to be too restrictive. Here's the line.

✓ BELONGS IN RAG

Past proposals and invoices — with prices, scope, customer names, dates. The single highest-value source for trade businesses.

Jobsite notes and field reports — what went well, what went wrong, surprises encountered. Lessons-learned lives here.

Materials and pricing reference data — in narrative form (e.g., "we typically charge $X per linear foot for trim, but bump to $Y for hardwood.")

Customer history and notes — preferences, past jobs, payment patterns, how they like to communicate.

SOPs and process documents — how you do walkthroughs, how you write quotes, how you handle complaints.

Vendor information — who's reliable, who's expensive, who has the best lead times for what.

Estimating rules of thumb — the heuristics in your head that aren't in any textbook.

✗ DOES NOT BELONG IN RAG

Live transactional data. "What's the current bank balance?" should query QuickBooks live, not a stale RAG snapshot. Same for active calendar availability.

Highly structured numbers needing exact recall. A materials catalog with exact unit prices belongs in a structured table (a database lookup), not free-text retrieval. RAG is good at "approximately right;" tables are good at "exactly right."

Sensitive personal data not meant for chat. Social Security numbers, full credit cards, employee health records. RAG retrieves; if you don't want it surfaced in a conversation, keep it out.

Real-time external feeds. Weather, stock prices, current Zillow listings. Use APIs and live tools, not stale RAG entries.

Anything you wouldn't want a junior employee to read. If it shouldn't be visible to the AI's questioner, it shouldn't be in the RAG store either.

The simple rule

Ask yourself: "*If a smart new employee read this, would it help them answer customer questions, write proposals, or make decisions consistent with how we run things?*" If yes, it belongs in RAG. If you'd rather they get the answer from a live system (QuickBooks, calendar, Stripe), it doesn't.

> **Common mistake:** dumping a 200-page master price sheet into RAG because it has prices. RAG will find *roughly* the right line, but a customer-facing quote needs the exact one. Lookup tables are better for exact numbers; RAG is better for the surrounding context ("we charge X for trim, but the Schulzes always negotiate down 10% so quote at Y").

Why your agent stack falls apart without it

Here's what happens, in concrete terms, when you skip the memory layer and just bolt a generic AI to your business processes:

Failure mode #1 — Wrong-but-confident pricing

Customer texts: "*What would you charge to install LVP in my 800sqft basement?*" Your AI agent — without RAG — answers: "*Typical pricing for LVP installation runs $5–8 per square foot, so your project would be $4,000–6,400.*" Confident. Articulate. Based on national averages it learned during training. **It's also not your number.** Maybe you charge $7.50/sqft because of the way you handle subfloor prep. Maybe this customer always gets your "neighbor rate" of $5.50. Maybe you're at capacity and your number for this week is $9 because you'd rather pass than rush. The AI doesn't know any of that. The customer accepts. You either eat the difference or come back to renegotiate. **One bad quote is worse than no quote.**

Failure mode #2 — Customer-history confusion

You've worked with the Schulz family for three years. Done their bathroom, two doors, a fence repair. They text your agent: "*Hey it's Mark, can we talk about the kitchen?*" Your agent — without RAG — treats Mark like a stranger. Asks for an address. Asks how he heard about you. Asks if he wants a free estimate. Mark doesn't reply. **You just damaged a 3-year relationship** because your agent had no idea who Mark was, even though Mark's history is sitting in your QuickBooks and your proposals folder.

Failure mode #3 — SOP ignorance

Your foreman has a rule: never agree to a job within 48 hours of a holiday weekend, because the suppliers can't deliver. Your AI agent — without RAG — doesn't know that. Books the job. Suppliers can't deliver. Customer is furious. **Your "AI assistant" just made a commitment your business can't keep** — not because it was malicious, because it had no memory of how your business actually operates.

These aren't theoretical. Each one happens within the first two weeks of bolting a generic AI onto a business with no memory layer. The technology works. It's just operating in the dark.

> IF YOUR KNOWLEDGE IS SCATTERED ACROSS QUICKBOOKS, YOUR FOREMAN'S HEAD, AND THREE SPREADSHEETS NOBODY ELSE CAN FIND, that's the exact problem Agent Logic helps small operators solve — turning what you already know into something an AI agent can actually use, without you learning four new tools. If that sounds like your situation, the closing page of this brief shows how I work with people on it.

> PATTERN: every "AI agent failure story" in trade businesses traces back to the same root cause — the agent had no orientation in the specific business it was supposedly running. The fix is never "smarter AI." The fix is "give it a memory."

Bad RAG vs Good RAG

Building a RAG system is easy. Building one that actually helps is not. The difference between a useful RAG and a useless one comes down to a few discipline points that most first-time builds get wrong.

Bad RAG

- **Dumps everything in.** Every email, every PDF, every random Word doc, every old contract from 2018. The system retrieves *something* for every question, but it's often the wrong something — an outdated price, a contradicted policy, a customer who fired you years ago.
- **No tags, no categories.** When the agent searches, it can't tell a proposal from an SOP from a personal note. Everything competes for relevance equally.
- **No citations.** The agent answers but won't tell you where the answer came from. If it's wrong, you can't trace it back. If it's right, you can't verify.
- **Never gets pruned.** Last year's pricing sits in there forever. Old vendor contacts. Discontinued product info. The agent keeps surfacing wrong-because-stale answers.
- **One workspace for everything.** Customer data, financial info, SOPs, jokes from a Slack channel — all in the same searchable pool. Permissions are an afterthought.

Good RAG

- **Curated sources.** Documents are vetted before they enter the knowledge base. Old/wrong/contradicted material gets pruned. The data going in is the data coming out — garbage in, garbage out applies harder to RAG than almost any other system.
- **Tagged and categorized.** Each document carries metadata: `type: proposal`, `customer: Schulz`, `date: 2026-04-15`. The retrieval system can filter by tag before it ranks by relevance.
- **Citations every time.** Every answer the agent gives includes a source pointer: "based on proposal-2026-04-15-Schulz.md, page 2." If the answer is wrong, you can find why. If it's right, you can hand the same source to a human reviewer.
- **Refreshed on a schedule.** New documents flow in regularly. Stale ones get retired. The knowledge base reflects the business as it is now, not as it was 18 months ago.
- **Workspaces split by sensitivity and audience.** Customer-facing answers pull from a "public" workspace; internal pricing decisions pull from a "private" one; financial and HR data live in workspaces with stricter access. Permissions are designed in, not bolted on.

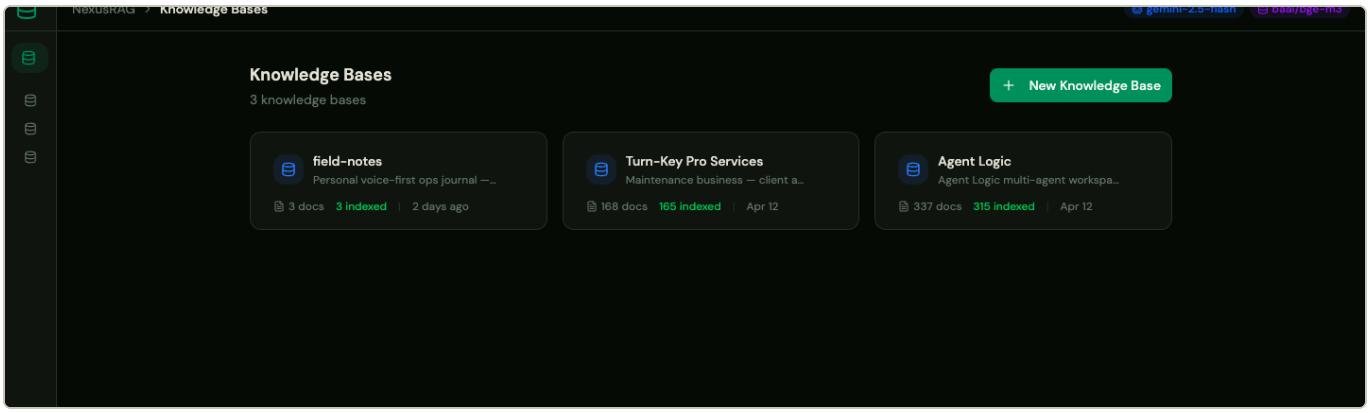
The discipline that separates them

The technical difference between bad and good RAG is small. The discipline difference is huge. Good RAG requires somebody (you, an assistant, an automation) to keep the knowledge base clean — sorting, tagging, pruning, refreshing — the same way you'd keep a real filing system clean. Skipping that work is the most common reason RAG projects "don't work." It's not the technology. It's the operational hygiene around it.

> **The non-negotiable:** if your RAG can't tell you *where* an answer came from, you don't have RAG — you have a black box that hallucinates with extra steps. Insist on citations from day one.

What it actually looks like in action

Concrete is more useful than abstract. Here's what one round-trip looks like in my own setup — the system I use every day to run my trade business. The system is called *NexusRAG*; the principles apply to any well-built RAG.



My actual NexusRAG dashboard, today: three workspaces (field-notes, Turn-Key Pro Services with 168 docs, Agent Logic with 337 docs), running on Gemini 2.5 Flash with BAAI/bge-m3 embeddings.

The scenario

I'm walking a property in Fond du Lac. A homeowner says, "wait, didn't you do flooring for the Schulzes last year? What did you charge them?" I pull out my phone and message my AI assistant via Telegram: "*What did we charge for the Schulz flooring?*"

The round-trip

> **Step 1 — Ask.** Telegram message: "What did we charge for the Schulz flooring?" hits my agent (a system I call SCT, but the principle is the same for any agent).

↓

> **Step 2 — Retrieve.** SCT calls NexusRAG's query endpoint with the question. NexusRAG searches across two workspaces (one for proposals/contracts, one for daily notes), runs a vector search plus a knowledge-graph lookup for "Schulz" as an entity, and reranks the candidates. Top result: a chunk from `proposal-2026-flooring-schulz.md` with the line-item pricing.

↓

> **Step 3 — Augment.** The retrieved chunk is added to the question as context. SCT now sees: "User question: what did we charge for the Schulz flooring? Relevant context: from proposal-2026-flooring-schulz.md (page 2): 'Hardwood install, 850 sqft @ $7.25/sqft = $6,162.50; subfloor prep $480; total $6,642.50.'"

↓

> **Step 4 — Answer.** SCT replies via Telegram: "Schulz flooring was $6,642.50 total — 850 sqft of hardwood at $7.25/sqft, plus $480 in subfloor prep. Source: proposal-2026-flooring-schulz.md (page 2)."

What just happened that wouldn't have without RAG

- **The number is real.** Not a guess based on national averages — the actual number we charged the actual Schulzes.
- **The breakdown is real.** Square footage, rate, prep cost — all from the actual proposal.
- **I can verify it.** The citation points me to the source document. If I'm in front of a customer, I can pull up that proposal and back up the answer.
- **Total time:** about 4 seconds end-to-end. Faster than I could find the proposal in my own files.

Without RAG, the same question gets a confident, articulate, made-up answer. With RAG, it gets the truth, sourced. That difference is what determines whether your AI agent saves you time or quietly destroys your customer trust.

> **THE LITMUS TEST:** ask your AI agent a question only your business can answer correctly. If the response is specific, sourced, and right — you have RAG. If it's plausible-sounding nonsense, you don't, and the rest of your stack is running on a

> foundation of guesses.

The cost reality

If RAG is the missing piece, what does it actually cost to set up? There are three real ways to make an AI "know your business," and the cost difference between them is not subtle.

Three approaches

Approach	How it works	Real cost (SMB scale)
Long context window	Stuff your entire business knowledge into every single AI prompt. Pay per token, every request.	$1,000–5,000+/mo at any real volume. Wasteful: you're paying to send the same data over and over. Also degrades in accuracy past about 1,000 tokens of irrelevant context, no matter what model marketing claims.
Fine-tuning	Retrain the AI model itself on your business data. The knowledge becomes part of the model's "training."	$5K–20K upfront for the training run, plus $40K–150K in labor to prepare clean training data. Locks you into a specific model. Has to be redone every time you switch models. Inference can be cheap, but the data-prep work is brutal at SMB scale.
RAG	Keep your data in a knowledge base. Retrieve relevant pieces at query time. Pay per query, not per document stored.	$500–2,000/mo if you use a managed service. Self-hosted is $0–50/mo for infrastructure. Scales with usage, not data size. Model-agnostic — switch models tomorrow, RAG keeps working.

What the numbers say for SMB scale

For a small business under $5M revenue with under 100 AI queries per day, RAG wins by a factor of 5–10x in cost-effectiveness. Not because it's a clever technology — because the alternatives are massively over-engineered for the volume. Fine-tuning is built for situations where you'll spend millions in inference and want to shave 10–15% off; that math doesn't apply to a flipper or a contractor. Long-context approaches make sense for situations where context changes every query; that doesn't apply when your business knowledge is roughly stable.

What you actually pay for in a RAG setup

- **The vector store** — where the searchable knowledge lives. Free options exist (ChromaDB, sqlite-vec); paid options run $10–100/mo at small scale.
- **The embedding step** — converting text into searchable form. Free if you use small local models; ~$0.02 per 1,000 documents if you use OpenAI or Voyage. Cents.
- **The retrieval + answer step** — whatever you'd already pay for AI calls. Adds ~10% on top because of the slightly longer prompt with retrieved context.
- **Your time keeping the knowledge base curated.** This is the real ongoing cost — an hour or two a week of pruning, tagging, and adding new documents.

For a contractor running 50 queries a day across a workspace of 1,000 documents, total RAG cost lands somewhere between $20 and $200 a month. The question isn't whether you can afford RAG. It's whether you can afford *not* to have it.

> IF IMPLEMENTATION IS WHERE YOU STALL — picking the vector store, choosing the embedder, wiring it into your existing tools, designing the workspaces, getting the citations right — that's where most first-time builds stop. Agent Logic ships pre-

> configured RAG installs as part of the DFY tier. We pick the stack, ingest your existing documents, set up the workspaces, and hand you a system that works on your real questions on day one. Details on the back page.

Common mistakes

Five mistakes that turn a working RAG system into a useless one. All of them are about discipline, not technology.

1. Treating RAG as a magic box

Symptom: feeding it everything you've ever written and expecting precision. Reality: RAG is a search system; if your data is messy, the search results will be messy. The investment isn't in the technology — it's in the curation of what goes in.

2. Skipping the curation work

Symptom: dumping a folder of 5,000 mixed PDFs and walking away. Reality: a good RAG starts with maybe 50–200 carefully chosen documents that represent your business well, and grows from there. Quality of input dominates quality of output. An hour spent pruning beats a week spent tuning.

3. Ignoring permissions

Symptom: one giant searchable pool. Anyone who asks the AI gets potentially anything back. Reality: split workspaces by sensitivity. Customer-facing pool (proposals, public marketing copy). Internal pool (margins, vendor relationships, pricing strategies). Restricted pool (HR, financials, legal). Permissions need to be designed in from day one, not bolted on after a leak.

4. Expecting perfect retrieval

Symptom: assuming RAG is always going to surface the right document. Reality: even a well-tuned RAG has a miss rate of 5–15% on tricky queries. For critical workflows (quotes, scheduling, contracts), build human verification into the loop. RAG accelerates the answer; a human still validates before money or commitment changes hands.

5. Not measuring whether it works

Symptom: setting up the system, watching it answer questions, and assuming everything's fine. Reality: pick 10–20 known-answer questions about your business. Run them through the system every month. Track whether it's getting them right. If accuracy drops, your knowledge base has drifted — time for a curation pass. Without measurement, you'll discover the system is broken only when a customer complains.

> **The biggest mistake of all:** running RAG without ever testing it. It will sound right even when it's wrong, because LLMs are fluent. Confident answers that aren't grounded in your data are worse than no answer at all — they erode the operator's trust in the system, which is the single most fragile asset in this whole equation.

Does your business need RAG right now?

Here's a five-question self-diagnostic. Answer honestly. If you say "yes" to three or more, RAG is the highest-leverage thing you could add to your operations stack this quarter.

1. **Have you tried ChatGPT or Claude on a real business question and gotten an answer that was confident, articulate, and wrong for your business?**
   If yes, you've experienced the missing-memory problem firsthand.
2. **Does most of the institutional knowledge in your business live in one or two people's heads (yours, your foreman's, your office manager's)?**
   If yes, that knowledge walks out the door if any of them leave. RAG is one of the few technologies that captures it before that happens.
3. **Have you re-researched the same question more than twice this year** (a vendor's quirks, a customer's history, last year's pricing for a similar job, the right way to handle a specific permit)?
   If yes, your knowledge base exists but it's not searchable. RAG fixes the search problem.
4. **Do new hires take more than 90 days to become genuinely productive in your business?**
   If yes, the orientation gap is huge — RAG can compress it dramatically by giving them the same searchable knowledge that the AI would use.
5. **Are you already running, or about to run, AI agents in any customer-facing workflow** (proposals, scheduling, follow-ups, intake, quoting)?
   If yes, you can't afford to skip RAG. Every customer interaction without it is a coin flip on accuracy.

Scoring

Yes count	What it means
0–1	RAG isn't your bottleneck right now. Focus on getting any AI agent into your workflow first; revisit RAG when the missing-memory pain shows up.
2	RAG would help, but the value is incremental. Add it after your next bigger operational improvement.
3	RAG is solving a real, recurring pain point in your business. Worth setting up in the next quarter.
4–5	You're losing real money to the missing-memory problem every week. RAG is the highest-ROI move on your operations roadmap. Start now.

How to start small

Don't try to RAG-ify your whole business at once. Pick *one* workspace — usually past proposals, because they have prices, customer names, and scopes — and ingest them. Test queries on questions you already know the answer to. Watch the accuracy. Then add the next workspace (jobsite notes, customer history, SOPs). The whole point of starting small is to build operational muscle for keeping the knowledge base curated — that's the part that breaks at scale, and it's better to learn the discipline on 100 documents than 5,000.

> **THE FIRST-WEEK TEST:** set up RAG on your past proposals folder. Ask it three questions you already know the answer to. If it gets all three right with citations, you have a working foundation. Build from there.

What's next

You finished the brief. You understand why your agent stack needs a memory, what to put in it, and what to keep out. The question now is what you do with that understanding.

Three places to go from here

Tier	What it is	When it's right for you
DIY	Pick a managed RAG service (Pinecone, Weaviate Cloud, Vectorize, ChromaDB) or self-host with open-source tools. Ingest your past proposals folder as your first workspace. Test on known-answer questions.	You have a few hours a week, you're comfortable with a small amount of technical setup (or you'll drive an AI assistant through it), and your operation is small enough that you can stay close to the curation work.
DWY (Do With You)	A workshop / cohort where you build out your business's RAG with guidance, alongside other operators doing the same. *Coming late 2026.*	You want company through the build. You're stuck on which documents matter, how to structure workspaces, or how to evaluate accuracy. You want to hear how other operators in similar businesses are doing it.
DFY (Done For You)	Custom RAG install. I work with you to identify your highest-value knowledge sources, set up workspaces with proper permissions, ingest your existing documents, wire it into your AI agents, and hand you a working system. Setup + monthly retainer for ongoing curation help.	Your time is worth more than figuring out vector stores. You want it running by next month. You'd rather have someone keep the knowledge base curated for you than try to do it yourself between jobs.

ABOUT ME. I'm Alex Jahn. I'm a working carpenter and house-flipper in Fond du Lac, Wisconsin. I've been building agent-based tooling for my own trade business since the beginning of 2026 — from day one of autonomous AI agents being practical inside real business workflows — and RAG is the load-bearing piece of every part of that stack. I run Agent Logic, where I help other tradespeople and small business operators get from "I tried ChatGPT and it didn't work" to "my AI agent actually knows my business and saves me hours every week." The unfair part of the pitch: no AI consultancy actually runs a real trade business. No carpenter actually runs a multi-agent stack with curated RAG. I'm the only person dumb enough to do both, which means I've already solved the problems you're about to hit.

How to reach me

If this brief was useful, even just to read, I'd love to hear it. If something in it was confusing or wrong, I'd love to hear that too — the failures are how I improve the next edition. Three ways:

Email	alexanderjahn79@icloud.com
Text	920-539-8814 — my actual cell, real human, no bot
Schedule a call	920-679-6207 — this number rings my AI assistant. Tell it you want time with me; it'll check my calendar and book the slot. You'll be talking to a RAG-powered agent the moment you call — that's the same kind of system this brief describes, in production. Consider it the demo.

The hardest part of an AI agent stack isn't the AI.
It's giving it a memory of your business that doesn't quit, take vacation, or forget.
Build it. Then build everything else on top.