The AI Contractor Trap

By Joe Giler

Table of Contents

Chapter 1: Read a Quote Line by Line: What Each Item Actually Costs to Build

The first time you get a quote for software work, you will feel one of two things. Either it is so high that you assume the contractor is trying to rob you, or it is so low that you assume you have found a bargain. Both reactions are usually wrong, and both come from the same place: you have no idea what any of the line items actually cost to produce.

That is the trap. Not fraud, not incompetence — information asymmetry. The person quoting you knows roughly how many hours each piece takes. You do not. Everything downstream of that gap — the scope creep, the surprise invoices, the "that wasn't included" conversations — flows from it.

This chapter is about closing the gap. Not so you can beat your contractor down on price. So you can tell the difference between a number that reflects real work and a number someone made up because you looked like you would pay it.

Why Quotes Are Deliberately Vague

Open a typical agency quote for an "AI-powered customer support chatbot" and you will see something like this:

- **Discovery & Requirements** — $4,000
- **AI Integration & Training** — $12,000
- **Custom Dashboard** — $8,000
- **Testing & QA** — $3,500
- **Deployment & Handover** — $2,500
- **Project Management (15%)** — $4,500

Total: $34,500. And you have absolutely no way to evaluate it.

Look at "AI Integration & Training — $12,000." What does that mean? It could mean three weeks of a senior engineer building a retrieval pipeline over your

documentation, evaluating chunking strategies, writing an eval harness, and tuning prompts against real customer tickets. It could also mean someone spent four hours wiring up an API call to a model provider and pasting your FAQ into a system prompt.

Both are honestly describable as "AI Integration & Training." One is worth $12,000. One is worth about $600.

Vagueness is not always malicious. Contractors write vague line items partly because they genuinely don't know the exact shape of the work until they're in it, and partly because specificity creates obligations. If the line says "AI Integration," they can deliver whatever they end up building. If the line says "RAG pipeline over 1,200 support documents with a chunk-level evaluation set of 100 labeled question/answer pairs, achieving 85%+ retrieval precision@5," they have to actually hit that.

Guess which one most contractors prefer to write.

The core move in this entire book: convert every vague line item into a specific, verifiable deliverable before you sign anything. You do not need to know how to build it. You need to know what "done" looks like well enough that a stranger could check it.

The Three Cost Drivers Behind Any Software Line Item

Every piece of software work decomposes into three cost drivers. Once you can see them, quotes stop being mystical.

1. Novelty. Has this been built ten thousand times before, or is it genuinely new? A login page with email/password and a reset flow is the most-solved problem in web development. There are libraries, templates, and hosted services that do it. Anyone charging you $6,000 for "user authentication system" in 2026 is charging you for a solved problem. Contrast that with "figure out why our model's answers get worse on invoices from one specific vendor" — that is genuinely novel work with an unknown time cost.

2. Integration surface. How many other systems does this thing have to touch, and do you control them? A feature that lives entirely inside one codebase is cheap. A feature that has to read from your 2009-era ERP, write to Salesforce, respect permissions defined in Active Directory, and stay in sync when any of the three

change — that is expensive, and the expense is mostly not the code. It is discovering how those systems actually behave, which is never what the documentation says.

3. Failure cost. What happens when it breaks? A marketing site that goes down for an hour is an annoyance. A payment flow that double-charges customers is a catastrophe. Higher failure cost means more testing, more review, more defensive engineering, more rollback planning. It is legitimately more expensive, and it should be. When a contractor quotes payment handling the same way they quote a contact form, that's a signal they haven't thought about it.

Whenever you see a line item you can't evaluate, ask the contractor to rate it on those three axes. "Is this a solved problem or new? What does it have to talk to? What happens if it breaks?" Their answer tells you more than the number does.

The Real Cost of the Twelve Most Common Line Items

These are working estimates for 2026, in the US/UK/EU/AU market, for a competent mid-to-senior freelancer or small agency. Rates vary enormously by geography, and offshore teams in South Asia, Eastern Europe, and Latin America often quote 40–70% lower. Treat every number as a range and an estimate, not a price list.

Assume a blended rate of $75–$175/hour for a competent independent contractor in a high-cost market, and $150–$300/hour for an established agency. Below $50/hour you are usually buying either offshore labor or inexperience. Above $300/hour you are buying a brand, a specialist, or a very short engagement.

1. User authentication (email/password, reset, sessions). Real cost: 4–12 hours if built on an existing auth provider, which it should be. Services like Auth0, Clerk, Supabase Auth, and Firebase Auth exist specifically so nobody has to build this again. Anyone building custom auth from scratch in 2026 either has a very specific compliance reason or is padding. **Fair range: $400–$1,800.** Add meaningful cost for SSO/SAML (enterprise single sign-on is genuinely fiddly — 20–40 hours), multi-tenancy, or granular role permissions.

2. Payment processing (Stripe or similar, one-time payments). Real cost: 8–20 hours. Stripe Checkout handles the hard parts. The work is in webhook

handling — making sure that when Stripe says "payment succeeded," your system reliably records it, exactly once, even if the webhook fires twice or your server was down. That's where the hours go, and it's where cheap implementations break. **Fair range: $800–$3,000.** Subscriptions with upgrades, downgrades, prorating, and dunning: add 20–40 hours.

3. "AI integration" — basic LLM API call. Real cost: 2–8 hours. Calling Claude, GPT, or Gemini from your backend, passing a prompt, returning a response, handling errors and rate limits. This is genuinely a small job. **Fair range: $200–$1,200.** If someone quotes five figures for "AI integration" and this is what they mean, walk.

4. RAG pipeline (AI that answers from your documents). Real cost: 40–120 hours for something that actually works. This is where honest AI project money goes. The work: ingesting and cleaning your documents (always messier than expected — PDFs with tables, scanned images, inconsistent formatting), chunking them sensibly, generating embeddings, setting up a vector store, building retrieval, handling the model's context window, and — critically — building an evaluation set so you can tell whether it's getting better or worse. **Fair range: $4,000–$18,000.** The evaluation piece is the one contractors skip and the one that determines whether you end up with a working system or a demo.

5. AI agent with tool use (books appointments, updates records, takes actions). Real cost: 60–200+ hours. Much harder than it demos. The gap between "works in the happy path" and "safe to let it touch production data" is enormous. You need permission boundaries, confirmation steps for destructive actions, audit logging, and extensive testing of failure modes. **Fair range: $6,000–$30,000+.** Be extremely suspicious of anything quoted under $5,000 here.

6. Fine-tuning a model on your data. Real cost: highly variable, 30–150 hours, plus compute. But the more important point: **you probably don't need this.** In 2026, for the vast majority of business use cases, good prompting plus retrieval beats fine-tuning on cost, flexibility, and maintenance. Fine-tuning earns its keep for specific output formats, narrow classification tasks, tone matching at scale, and latency/cost reduction on high-volume simple tasks. If a contractor leads with fine-tuning for a general "AI assistant," ask them directly why retrieval isn't sufficient.

Their answer will tell you whether they know what they're doing. **Fair range if genuinely needed: $3,000–$20,000** plus ongoing compute.

7. Admin dashboard (view, search, filter, export data). Real cost: 20–60 hours depending on how many entity types and how much custom logic. Tools like Retool, Appsmith, and Forest Admin can produce a competent internal dashboard in a fraction of custom build time. **Fair range: $2,000–$9,000 custom; $500–$2,500 on a low-code tool plus subscription.** Always ask whether a low-code admin tool was considered. If the answer is "we prefer to build custom," ask what specifically requires custom.

8. Mobile app (iOS + Android, moderate complexity). Real cost: 300–900 hours for a real product. Cross-platform frameworks (React Native, Flutter) cut this meaningfully versus two native codebases. Add time for app store review, which is a process, not a step. **Fair range: $25,000–$120,000.** Anyone quoting $8,000 for "a mobile app" is quoting a wrapper around your website, which may be fine — but you should know that's what you're buying.

9. "Discovery" or "Requirements Gathering." Real cost: 8–40 hours. This is legitimate work — understanding your business, mapping the process, writing specs. It is also the easiest line to inflate because its output is a document, and documents can be any length. **Fair range: $1,000–$6,000.** Insist on knowing what the deliverable is: a written spec? Wireframes? A data model? "Discovery" with no named artifact is a $4,000 conversation.

10. Testing & QA. Real cost: typically 15–30% of development time when done properly, often folded into the dev estimate rather than separate. Seeing it as a separate line isn't a red flag — but a suspiciously round number that's exactly 10% of the project total suggests it was calculated as a percentage, not estimated as work. **Ask what kind of testing.** Automated unit tests? Integration tests? Manual click-through? Load testing? These cost wildly different amounts and deliver wildly different value.

11. Project management. Commonly 10–20% of project total. This is normal industry practice and not inherently a scam. On a $30,000 project with multiple people, someone genuinely has to coordinate. On a $8,000 project delivered by one

freelancer, a 15% "project management" fee is that person charging you to email you. **Rule of thumb: PM charges make sense above roughly $20,000 or with three-plus people involved. Below that, question it.**

12. Deployment & DevOps. Real cost: 4–16 hours for a standard app on a modern platform (Vercel, Railway, Render, Fly.io, or a managed cloud service). 40–150 hours if you need custom infrastructure, Kubernetes, multi-region, or compliance-driven isolation. **Fair range: $400–$2,000 standard; $4,000–$20,000 for genuine infrastructure work.** The tell: ask what platform. If the answer is "AWS" with no further detail on a small project, you may be paying for complexity you don't need.

A note on all of the above: these are estimates drawn from typical market ranges, not survey data. Rates move, and your specific situation may legitimately fall outside these bands. Use them as a starting point for conversation, not a weapon.

The Arithmetic Test: Does This Quote Describe a Human Being?

Here is the single most useful thing you can do with any quote, and it takes five minutes.

Divide the total by a plausible hourly rate, and see how many hours that implies. Then ask whether that many hours matches the described work.

Take the $34,500 chatbot quote. At $150/hour, that's 230 hours. At $100/hour, 345 hours. That's between six and nine weeks of one person working full time, or three weeks of a three-person team.

Now ask: does building a customer support chatbot over your existing documentation, with a dashboard, plausibly take six to nine full-time weeks? For a serious production system with real evaluation, integrations into your ticketing platform, and handoff-to-human logic — maybe. For "chatbot on our FAQ" — absolutely not.

Run this in both directions. A $4,000 quote for the same system implies 27–40 hours. That's one person for one week. You are being quoted a prototype. Which might be exactly what you want! But you should know it, and you should not be surprised in month three when it can't handle load.

Ask the contractor directly: "How many person-hours does this quote represent, and how are they distributed across the line items?"

This question is completely reasonable and most clients never ask it. The responses you get:

- **A clear breakdown** — great sign. They've thought about it and they're comfortable showing their work.
- **"We price by value, not hours"** — sometimes legitimate, especially for specialists delivering outsized business impact. But it's also the standard deflection. Follow up with: "Understood — I'm not trying to renegotiate the rate, I just want to understand the scale of the work. Roughly how many hours?" If they still won't say, that's information.
- **Visible discomfort or vagueness** — they either haven't estimated it or the number is embarrassing.

The Line Items That Should Make You Stop

Certain phrases in a quote warrant an immediate follow-up question. Not because they're always dishonest — because they're always ambiguous.

"AI model training" on a project where you have no labeled data. Training requires examples. If you haven't given them thousands of labeled examples, and they haven't quoted for creating them, ask what exactly is being trained. Often the honest answer is "prompt engineering," which is real work but costs a fraction as much.

"Custom AI model." In 2026, almost nobody outside of well-funded research labs should be training a custom foundation model. When contractors say "custom AI model" they usually mean a fine-tune, a prompt, or a retrieval system. Ask which. The word "custom" is doing enormous load-bearing work in that phrase.

"Proprietary framework" or "our internal platform." This means the deliverable may not be portable. If they build on their own framework, you may be unable to hire anyone else to maintain it. Sometimes this is fine and the framework is genuinely good. Often it is a lock-in mechanism. Ask: "If we parted ways, could another developer take over this codebase? What would they need to learn?"

"Ongoing optimization" with no defined scope. A recurring charge for unspecified improvement is a subscription to vibes. Convert it into either a defined maintenance SLA (response times, uptime, bug fix turnaround) or a fixed block of hours per month you can direct.

"Setup and configuration" as a large line. Configuration of what? Setting up a hosting account and a database is a few hours. If this line is thousands of dollars, something specific is being configured and you should know what.

Any line item over $5,000 with fewer than fifteen words describing it. The ratio matters. Big numbers deserve detail.

How to Ask for a Rebuild of the Quote (Without Blowing Up the Relationship)

You are not trying to insult anyone. You are trying to buy something you can evaluate. Most good contractors will respect this; the ones who bristle are telling you something useful.

Here is language that works. Adapt it to your voice.

"Thanks for this — the overall approach makes sense to me. Before I sign off, I'd like to get the quote into a form where I can track progress against it. Could you break this into deliverables of roughly $2,000–$5,000 each, with a one-sentence definition of 'done' for each one? I'm not looking to renegotiate the total, I just need to be able to tell where we are mid-project."

Notice what that does:

- **It doesn't challenge the price.** You explicitly say you're not renegotiating. This removes the defensive reaction.
- **It creates milestones.** $2,000–$5,000 chunks give you natural checkpoints and natural exit points.
- **It forces definitions of done.** This is the whole game. A deliverable with a definition of done is a deliverable you can accept or reject.
- **It gives a business reason.** "I need to track progress" is unarguable. "I think you're overcharging me" starts a fight.