Streamline AI

Should you *build an AI intake tool* or buy one?

Re-examining the build vs. buy question in the era of AI agents and AI coding tools

Contents

Build vs. Buy

Building an AI-powered legal intake and matter management solution looks easy. *But is it?.*

AI is everywhere, and it's fundamentally changed the way we think about the world, including how we decide whether to build or buy a software solution.

It's true that AI coding tools have made it faster than ever to ship a working prototype. They've also made it easier to mistake a prototype for a fully functioning product, especially when that product will hold your team's most sensitive work.

The prototype illusion

Building a demo is easy. *Creating a fully functioning product isn't.*

Tools like Claude, Copilot, and Cursor have genuinely compressed the time it takes to go from an idea to a working prototype. A form that captures intake requests, a Slack bot that routes questions to a channel, a template that generates a first-draft NDA: you can stand this up in days. That's real progress, and it's changed how teams think about what's possible.

But the shortcut is narrower than it looks. Getting to a demo used to represent 60 to 70% of the effort. With AI assistance, it's closer to 20%. The remaining 80% of the effort required to get a working solution in place, including security, permissions, integrations, and compliance, hasn't actually gotten easier. It's just become easier to underestimate.

INITIAL BUILD:

20%

The prototype phase. Moves quickly with AI assistance. Looks and feels like most of the work.

PRODUCTION READY:

80%

Security, permissions, integrations, edge cases, maintenance. The work is less visible, and it doesn't go away.

What AI can't shortcut

The features and functionality that legal teams need most *are exactly the areas that AI-assisted coding handles worst.*

AI assistants like Claude are designed as individual productivity tools. A single user can draft, redline, or route a request to the next person in a chain, and that is roughly where native capability ends. There is no out-of-the-box way for a manager or GC to see all their team's requests in one view, no way to control who has access to what, and no audit trail for your compliance function to rely on. Those are not features you build on top of an AI assistant. They are the foundation the tool has to sit on before it can function as the operating system for an entire team.

Everything that makes a tool work as a shared system has to be built deliberately, and no amount of faster coding eliminates that work.

Knowing who can see what

Access controls sound like an IT concern, but they are a legal concern first. Before you can route, review, or store a single request, someone has to define who has access to which matters, how new team members are added, and how all of that activity gets recorded. Get any of it wrong, and you've created real exposure: privilege breaches, confidentiality risks, and compliance gaps that tend to surface at the worst possible time.

Getting requests to the right people

Routing a matter from submission to resolution sounds straightforward until you account for everything that happens in between. Different request types go to different reviewers. Some need sign-off from multiple people before moving forward. Deadlines slip, and someone needs to be notified. None of that complexity is unique to legal, but all of it has to work reliably when it is handling sensitive legal work, and building it to that standard takes considerably more than standing up a form.

Producing documents you can stand behind

Legal output is not like other business output. An NDA or a contract has to say the right thing every time, not approximately the right thing most of the time. That means the system generating it cannot rely on AI output that varies by run. The clause logic, the defined terms, and the exhibit numbering: all of it needs to be consistent, auditable, and reproducible, because the documents it produces may eventually need to hold up in diligence or discovery.

Keeping connections working over time

Connecting a new tool to the systems your team already uses is a few days of work. Keeping those connections working as vendors update their platforms, change their authentication requirements, and deprecate old functionality is an ongoing commitment that extends for as long as the tool is in use. Most teams account for the first part and underestimate the second. When these changes inevitably happen, your team has to go back to the drawing board and start over with a new round of development.

Delivering AI that actually works in production

Showing that AI can read a request, extract key information, and send a follow-up is easy to demonstrate. Deploying that capability reliably across every variation in how requests actually arrive is a different project. The gap between a convincing demo and a real, working solution is not a matter of polish. It is a matter of building the infrastructure that keeps the AI behaving correctly when things do not go according to plan.

Giving leadership visibility into what's happening

Once you're using a tool, the questions start coming quickly. How many NDAs are open? Where are things stalling? Which matters are overdue? The ability to answer those questions without filing an engineering request, or waiting for someone to manually pull a report, is not a luxury. It's what makes the tool useful to anyone beyond the person submitting requests.

The application sprawl conundrum

When anyone can create an application, *who's tasked with making sure they're all compliant and work together?*

When AI coding tools make it easy for anyone to spin up an application, the natural result is that many people do. A paralegal builds an intake form. Someone in legal ops connects it to a Slack channel. A senior associate adds a separate tracker for NDAs. Each piece works in isolation, and each one gets built with good intentions. But none of them were designed to talk to each other; none of them were built with a shared understanding of who should have access to what.

The organization now has several half-connected tools that each capture part of the workflow, a security posture that nobody has formally reviewed, and no single person who owns any of it. When something breaks, a new hire needs access, or legal needs to produce records for diligence, the question of which tool holds the authoritative version of anything becomes nearly impossible to answer.

There are no common permissions, no unified audit trail, no consistent way to search across matters or pull a report that reflects the full picture. What looks like progress—a team actively solving its own problems—quietly becomes its own problem.

True cost of ownership

The build is 20 to 30% of what you'll spend. The rest comes after.

The well-documented industry pattern for internal tools is that the initial build accounts for roughly 20 to 30% of the total cost of ownership over a three- to five-year horizon. The rest is everything that follows.

The other 70 to 80% is made up of ongoing maintenance as legal's needs evolve, fixes every time a vendor updates a platform your tool connects to, compliance changes as new regulations come into effect, and a steady stream of requests from internal stakeholders that never fully stops. Each one is small in isolation. Together they add up to a significant and largely invisible ongoing commitment.

Internal engineering teams are good at building. Structurally, they are less well-suited for the work that follows. Every update, no matter how small, has to compete for time against the work that drives revenue. Legal requests tend to lose that argument. Over time, the tool gets quietly worked around, and the team drifts back to spreadsheets and ad hoc messages for anything it does not handle cleanly.

The failure mode is rarely dramatic. Tools built internally do not usually get formally shut down. They gradually stop being maintained, the team works around the broken parts, and eventually someone new arrives, looks at the situation, and decides to start over.

Tech companies: The opportunity cost hiding in plain sight

Every resource spent on internal legal tooling is a resource not spent on your core business.

The most consistently underestimated cost in build vs. buy decisions is not the engineering time. It is everything that surrounds it. Defining requirements. Aligning stakeholders. Managing the people doing the work. Reviewing what they produce. And eventually, if the goal is a tool that works for the whole team rather than a collection of individual workarounds, building the infrastructure that makes that possible.

None of that shows up as a line item. It shows up as work that got delayed, decisions that took longer than they should have, and attention that was pulled away from the things that actually drive the business forward.

What actually changes with a purpose-built solution

The value is in what comes bundled with the software.

A purpose-built legal intake solution ships with the things that take the longest to get right: the access controls, the audit trails, the integrations, and the workflow logic designed around how legal teams actually operate. Updates, security patches, and compliance changes are handled on a predictable schedule, without competing for internal capacity.

The case for buying is not that building is impossible. It is that the ongoing investment required to do it well is larger, less visible, and harder to scope than it appears at the start. A purpose-built solution converts an open-ended internal commitment into a single, predictable line item that scales with your business.

Why Choose Streamline AI

The Honest Math

At first glance, building a custom intake and matter management solution—especially with today's AI tools—can seem like the faster, cheaper option. But when you take hidden costs, business needs, legal expertise, and continuity risks into account, the choice remains as clear as it's ever been: choosing a vendor like Streamline AI is the easiest, most cost-effective choice.

	HOMEGROWN SOLUTION	Streamline AI
Time to first value	6-9 months to MVP (even with AI tools and off-shore contractors)	**4 weeks to live**
Year 1 build cost	$200-$275k (1 senior engineer for 6-9 mos)	**$0 build cost**
Software and infrastructure cost	$10-20k (APIs, infra, security)	**Included in license**
Engineering FTE drag	.25-.5 FTE in perpetuity	**Zero engineering load**
Legal SME time to scope	200+ hours	**Standard implementation**
3 year TCO	$400-600k	**Licensing x3 years**
Roadmap velocity	Competes with product roadmap	**Continuous, every release**
Continuity risk	Engineer leaves, app orphaned	**Vendor-supported, transferable**

Streamline AI vs. a Homegrown Solution - A Technical Deep Dive

	Streamline AI	HOMEGROWN SOLUTION
Access and permissions		
Role-based access controls	Included. Managers and GCs see their team's requests; individual contributors see only their own. Configured at setup.	Requires custom architecture before the tool is usable as a team system. Significant engineering work, not an add-on.
Role-based access controls	Centralized provisioning and deprovisioning your IT team can manage directly.	Has to be designed and built from scratch. Often skipped in early versions, creating security gaps as the team grows.
Audit trails	Full activity logs included. Holds up in diligence and litigation discovery.	Requires deliberate instrumentation. Rarely built correctly in initial versions and difficult to retrofit later.
Attorney-client privilege protection	Access controls designed specifically to protect confidential legal communications.	Depends entirely on how permissions are implemented. Privilege breach risk if scoped incorrectly.
Workflow and routing		
Conditional routing	Built-in logic routes requests based on type, risk level, urgency, and team structure.	Requires building a state machine. Common point of failure for internal builds.
Multi-step approvals	Parallel and sequential approval flows configured without engineering involvement.	Custom-built per workflow. Each new approval type requires additional development.

SLAs and deadline tracking	Automated deadline tracking with escalation rules and reminders.	Requires custom logic for each rule. Often deprioritized in early builds and added reactively.
New contract types	Added via configuration. Legal team can request changes without an engineering ticket.	Every new contract type requires a development cycle. Becomes a recurring prioritization conflict.
Integrations		
CLM integration	Pre-built, maintained integration. Survives vendor API changes.	Built once, maintained indefinitely. Each vendor update is an unscheduled engineering task.
New contract types	Included with auth flow management and version compatibility handled by Streamline.	Custom integration requiring ongoing maintenance through DocuSign and similar vendors' auth changes.
Slack and messaging	Signed event handling, rate limiting, and webhook security managed and updated by Streamline.	Easy to build initially. Slack ships breaking changes regularly, requiring recurring engineering attention.
AI and automation		
Intake automation	Reads inbound requests, extracts fields, follows up on missing information, handles attachments, maintains context.	Prototyping the happy path is fast. Production-grade stateful agent behavior requires guardrails, evals, fallback paths, and observability tooling.
User experience for legal personas	Designed specifically for how legal teams operate. UX reflects the expectations of GCs, AGCs, and legal ops.	Generic tooling adapted for legal use. The last mile of fit for legal workflows has to be built and iterated on internally.

Reporting and visibility		
Dashboards and reporting	Built-in reporting for volume, SLA performance, matter status, and team workload. No SQL required.	Requires custom development for each report type. Every new report from a stakeholder is an engineering request.
Search	Full-text search across matters, requests, and documents included.	Requires indexing and search infrastructure. Often deferred and difficult to add retroactively.
Maintenance and ownership		
Security patching	CVEs and dependency updates handled by Streamline. No internal engineering required.	Falls to internal team. Unscheduled, recurring, and competes with product roadmap work.
Compliance updates	Streamline tracks regulatory changes and updates the product accordingly.	Internal team responsible for identifying and implementing compliance-driven changes.
Knowledge continuity	No dependency on individual engineers. Support and documentation maintained by Streamline.	When the engineer who built the tool moves on, every subsequent change becomes a re-learning cost.
Time to value	Weeks to full implementation.	3 to 4 months to a usable MVP; 6 to 9 months to a stable product.
Cost structure	Single predictable annual line item covering software, maintenance, support, and updates.	Initial build is 20 to 30% of total cost. The remaining 70 to 80% is maintenance, iteration, and ongoing ownership.