

Eleven businesses you could build on a model that only makes decisions.

Jev doesn't write for you. You hand it context and a set of options, and it hands back a classification, a score, or a probability. That's a narrower product than a chatbot, and the narrowness is the entire opportunity. Eleven places I'd look, and four questions that generate more of them than any list.

BY DAVE BROWN

Almost everything built on language models in the last three years was built around generation. Write the email, draft the post, summarise the meeting, produce the paragraph. That's the part that demos well, which is why it's the part that got built, and it's also the part where every product now looks like every other product.

A decision model points somewhere else. It points at the layer of software nobody demos: the thousands of small judgments a product makes before it shows anybody anything. Is this listing spam. Is this lead worth a call. Is this customer about to leave. Is this paragraph relevant. Which link do I click next. Every piece of software has that layer, and in most companies it's one of three things: rules somebody wrote in 2014 and nobody has touched since, a person clicking through a queue, or a frontier model being handed a thousand tokens of context to eventually produce the word **yes**.

There are two capabilities worth holding in your head. The first is that Jev classifies at volume and returns probabilities rather than verdicts. The second is that Jev-Ultrafast turns those decisions into browser actions, which in plain terms means it's good at using your browser to do things. The first capability attacks judgment work. The second attacks clicking work. Most of what follows is one or the other, and the two ideas I'd start with sit at opposite ends of that split.

THE MENTAL MODEL EVERYTHING ELSE FOLLOWS FROM

Stop asking what you can build *with* a decision model and start asking where a decision is currently too expensive to make often. That's a different search, and it turns up businesses instead of features. Expensive has three forms: a person doing it by hand, a rule doing it badly, or a big model doing it correctly at a price that forces you to sample.

Sampling is the tell. Any time a company reviews 1% of its calls, spot-checks a fraction of its listings, or scores leads on four fields because scoring them on forty was never affordable, there is a product in the gap between what they check and what they'd check if checking were free.

YES

The entire useful output of an enormous number of frontier-model calls running in production right now.

1% → all

Sampled review becomes complete review. That change, not the model, is what customers buy.

73%

A probability instead of a verdict. Thresholds are how software acts on uncertainty without a human in every loop.

Browser

Jev-Ultrafast's second act: decisions that become clicks, on the thousands of sites that never shipped an API.

Two notes on how to read this. First, none of these ideas are secret, and the idea was never the hard part, so for each one I've tried to name where the defensibility actually sits, because in every case it's somewhere other than the model. Second, I've deliberately quoted no prices, no latency figures and no accuracy numbers. The argument for these businesses is an economic one, the economics move monthly, and a number I write today is a liability by the time you read it. Check the current pricing yourself and run your own evaluation before you build a company on a cost assumption.

WHAT'S IN HERE

- 01 The shape of the opportunity. Two capabilities, four categories, and the one product not to build with this.
 - 02 Seven ideas that use the model differently. Shopping, browser labour, document search, agent safety, marketplace quality, churn, and competitive intelligence.
 - 03 Four more, kept short. Lead scoring, an injection firewall, a model router, and the inbox.
 - 04 Finding your own. Four questions, a one-page spec to fill in before you build anything, and a table of all eleven.
 - 05 Where I'd be skeptical. The three ways this category of business dies, said plainly.
-

PART ONE · THE SHAPE OF THE OPPORTUNITY

Two capabilities, four categories, and one thing not to build.

Worth five minutes before the list, because the categories are what let you tell a real idea from a demo. Every idea further down is one of these four moves applied to a specific industry.

The first capability is mass classification with calibrated output. You give it context and the possible answers; it gives you back which one, and how sure. The second is that the same decision loop can drive a browser: look at the page, decide the next action, take it, look again. Neither is a new idea. What's new is the price of doing it a million times, and price is what determines whether something is a feature in somebody's roadmap or a company.

CATEGORY	WHAT THE MODEL IS ACTUALLY DOING	WHAT YOU'RE REPLACING
Browser automation	One decision per page state, turned into one click. Repeat a few hundred times.	A person with forty tabs open, doing the same route through the same ugly portals every morning.

CATEGORY	WHAT THE MODEL IS ACTUALLY DOING	WHAT YOU'RE REPLACING
Classification at volume	A judgment per chunk, per message, per listing, precomputed and stored as metadata.	Search that only matches words, or a room of reviewers, or a rule set nobody trusts.
Customer and market intelligence	The same few judgments applied to every interaction instead of a sample of them.	A quarterly spreadsheet, a sentiment average, and somebody's intuition about why people left.
Safety and routing	A cheap decision placed in front of an expensive one, or in front of an irreversible one.	A frontier model used as a bouncer, a hand-maintained rule table, or nothing at all.

The fourth row is the one I'd point at hardest. A decision model is unusually well suited to sitting in front of other software: deciding which model handles a request, whether an action is safe to execute, whether a piece of retrieved text should be trusted. That position is valuable and uncomfortable in equal measure, and I come back to why further down.

THE THING NOT TO BUILD

Don't build a product whose value is prose. If your demo ends with a paragraph on screen, you've picked the wrong model, and you'll spend a year explaining why your writing is worse than the thing everyone already has a subscription to. The output of these businesses is a decision, a score, a queue, a filled-in cart, a filtered list, an alert. Nothing you'd read for pleasure.

And don't build "an AI that uses the internet." That's a capability, not a business. Every version of that pitch I've seen dies in the gap between a good demo and one specific customer's messy workflow. The narrow version, one industry, one route, one artifact delivered every morning, is less exciting and considerably more likely to get paid.

If the expensive part of your product is producing the word yes, you don't have a language problem. You have a pricing problem, and pricing problems are the ones that turn into companies.

Seven ideas that use the model differently, not seven flavours of one idea.

Ordered roughly by how quickly a small team could get to something a customer would pay for. Each one has a wedge, a buyer, and a hard part, and the hard part is never the model.

01

Let people shop by describing the thing, and let the model do the clicking.

BROWSER AUTOMATION · CONSUMER · JEV-ULTRAFAST

A conversational layer over ecommerce sites that the merchant never agreed to integrate with. The customer says "I need something for a wedding," then "less formal," then "green," then "under \$100," and the system actually operates the site: runs the search, clicks the filters, opens products, picks variants, changes sizes, compares two options, keeps going when the answer isn't right yet.

Navigating a store is a stream of very small decisions. Which filter next. Which result to open. Which variant matches what they said. Is this a better fit than the last one. Should I keep looking. Ask a frontier model each of those and a single shopping session can cost more than the merchant's margin on the sale, which is the real reason these assistants have stayed demos. Change the cost of one decision and the session becomes affordable.

I'd start as a browser extension in one category rather than a universal shopping agent, because the refinement language is domain-specific ("less formal" means something precise in apparel and nothing at all in building supplies), and because variants are where generic agents fall over. Sizes, colours, stock, the difference between a product that exists and a product page that exists. Get one vertical genuinely good and the second is mostly a new vocabulary, not new software.

WHAT MAKES THIS HARDER THAN IT LOOKS

Shopping isn't search. A lot of what people want is taste, and taste is the part a filter-clicking agent has least access to. The honest early version is a very good narrowing tool that hands a human three candidates, not a system that decides.

Then the business model. Affiliate revenue is the obvious answer and it's fragile, because you're dependent on attribution surviving a session your own software drove. Stop before checkout in version one: fill the cart, hand it back. Paying with someone else's card on a site that never agreed to any of this is a different product with a different risk profile, and it can wait.

THE WEDGE

Extension, one category, no merchant integration.
Narrowing, not deciding.

WHO PAYS FIRST

Affiliate at consumer scale, or a merchant who wants it on their own site as a licensed search layer.

THE HARD PART

Variants, stock accuracy, bot defences, and attribution surviving an agent-driven session.

02

Pick one job that is mostly clicking, and sell that job done.

BROWSER AUTOMATION · B2B · JEV-ULTRAFAST

The filter I'd use: find an expensive employee who spends an unreasonable share of the day clicking around websites. Checking a hundred supplier sites for price and stock. Searching government portals for relevant tenders. Pulling claim statuses out of insurance portals. Matching twenty buyer requirements against new listings every morning. Chasing permit records through municipal sites that look like they were built in 2003, because they were.

Each of those runs is hundreds of trivial decisions: click this, search that, take this result, skip this page, copy this field, move to the next site. Historically a model-driven agent made each of those decisions slowly and at a cost that ruled out the whole workflow. That's the constraint being relaxed, and this is the category where I'd expect the first unglamorous, profitable companies.

The mistake is selling an agent. Nobody wants an agent; they want the sheet. Sell the artifact: the pricing table in their inbox at 7am, the queue of new tenders that match their capability statement, the daily status file that used to take a coordinator until lunch. The agent is your implementation detail, and keeping it that way also means a broken portal is a delivery problem you absorb rather than a feature the customer watches fail.

WHERE THE DEFENSIBILITY ACTUALLY SITS

Not in the model, and not in your prompts. It's in the accumulated map of two hundred badly built portals: which ones paginate strangely, which ones rate-limit, which ones log you out at midnight, which field is the one that actually matters. That map takes months to build and nobody who buys from you will ever rebuild it. Around it sit the boring things enterprises pay for: credential handling, an audit trail of every page touched, and a number you commit to for freshness.

THE WEDGE

One workflow, one industry, delivered as a file or a queue on a schedule.

WHO PAYS FIRST

The manager of the person doing it now. Price against the fraction of a salary, not per API call.

THE HARD PART

Logins, MFA, terms of service, and silent breakage. You need to know you failed before the customer does.

03

Search what a document means, by deciding what it means in advance.

CLASSIFICATION AT VOLUME · B2B · PRECOMPUTED JUDGMENTS

Semantic search is good at finding related words. Ask for "car" and you'll get "automobile." It's much weaker when the question requires interpretation: *show me every paragraph describing something that could cause this car to break down*. No embedding of that sentence reliably finds the paragraph about a coolant seal.

So classify first. Run every chunk of the corpus against a set of questions (does this discuss pricing, does this create legal exposure, does this name a competitor, does this contain a deadline, does this contradict something said elsewhere, does this describe a defect), and store the answers as searchable metadata. Now a user searches judgments instead of words. When somebody invents a new question six months in, you rerun the corpus against that one question rather than rebuilding anything.

The markets are the obvious ones: discovery, diligence, regulatory review, research libraries, call transcripts, whatever a company calls its knowledge base. The product, though, is the question library. A firm's hundred and forty questions, refined over two years of matters, versioned, with a measured accuracy per question, is a real asset and a very sticky one. An index is not.

THE NUMBER YOU HAVE TO BE ABLE TO SAY OUT LOUD

In discovery and compliance, a missed paragraph isn't a worse search result, it's an exposure. That means recall, per question, measured against a labelled set, plus the ability to tell a partner or a regulator what your recall was and how you know. Precision keeps users happy; recall is what makes the product usable in the rooms where the money is. Budget for the labelling work from day one, because it's the part that can't be bought cheaply and the part competitors can't copy from your marketing site.

THE WEDGE

One corpus type with a painful recurring question set. Diligence and regulatory beat "all documents."

WHO PAYS FIRST

Whoever is currently paying people by the hour to read for one specific thing.

THE HARD PART

Measured recall per question, chunking that doesn't split the meaning, and reruns that stay affordable.

04

A permission layer between an agent and the real world.

SAFETY AND ROUTING · INFRASTRUCTURE · EVERY TOOL CALL

An agent is about to do something consequential: issue a refund, cancel an account, delete a table, change billing, send a sensitive email, alter permissions, approve a purchase. Put a cheap decision in front of it. Is this financially consequential. Is it reversible. Is the user's intent clear enough to justify it. Does it conflict with a stated policy. Is the agent unusually uncertain right now. Then act on the answer: execute, ask a clarifying question, require a human, or block.

The reason this needs a cheap decider is arithmetic. Agent systems make a great many calls, and using a frontier model to check each one doubles your latency and adds a cost that scales with exactly the thing you were trying to automate. A safety layer that costs as much as the work isn't a safety layer, it's a tax nobody pays.

The pitch writes itself, which should make you suspicious rather than confident. You're asking to sit in the hot path of somebody else's product, which means you own their latency and, on your worst day, their outage. Worse, a false block is far more damaging to adoption than a false allow: one blocked legitimate refund and an engineer turns you off permanently.

HOW I'D ACTUALLY SEQUENCE IT

Ship as observability first. Score every action, block nothing, and give the team a weekly list of the riskiest things their agents did while nobody was watching. That report is easy to say yes to, it produces the labelled data you need, and it earns you the right to be in the path later. Selling a blocking product on day one means arguing about your false-positive rate before you have one to quote.

THE WEDGE

Passive scoring and an audit log. Enforcement only after the customer trusts the scores.

WHO PAYS FIRST

Teams whose agents already touch money, customer records, or infrastructure in production.

THE HARD PART

Being in the hot path, false blocks, and the fact that frameworks may ship a version of this for free.

05

Moderation that returns a probability, with thresholds you can move on a Tuesday.

CLASSIFICATION AT VOLUME · MARKETPLACES · PROBABILITY OUTPUT

Any platform with user-generated listings (marketplaces, job boards, real estate, events, directories, creator platforms) needs two things from every submission. Is it allowed, and is it any good. Prohibited, fraudulent, misleading, policy-violating on one side; enough detail, clear title, useful description, appropriate images, not spam, not a duplicate on the other.

What changes the workflow is the probability. Instead of **good / bad**, you get **92% likely to meet the quality bar**, and thresholds become a business decision rather than an engineering one. Above 95%, publish. Between 70 and 95, ask the seller to improve it. Below 70, route to a human. When the platform raises its standards, you move the numbers and rescore the catalogue rather than retraining anything.

A note on who to sell to. Trust and safety is a cost centre with an incumbent vendor and a procurement process. Listing quality belongs to whoever owns conversion, and that person has a revenue number they're measured on and far more freedom to try something. Same engine, same classifications, dramatically different sales cycle. Lead with quality, and let safety be the thing you expand into once you're already scoring every listing.

THE WEDGE

Quality scoring for one marketplace vertical, sold on conversion rather than compliance.

WHO PAYS FIRST

Mid-sized marketplaces past the point where founders read listings and short of a T&S org.

THE HARD PART

Calibration you can defend, appeals, and the fact that thresholds without measurement are guesses.

06

Score every turn of the conversation, then look at what happened just before it went bad.

CUSTOMER INTELLIGENCE · B2B · PER-MESSAGE CLASSIFICATION

Ingest every customer interaction (tickets, chats, emails, call transcripts, sales and success calls) and classify each turn rather than each conversation. Frustration, confusion, satisfaction, escalation risk, churn risk, purchase intent, urgency. One number per customer tells you almost nothing. A trajectory tells you a lot: 10% frustrated, 18, 31, 68, 87.

The question that trajectory lets you ask is the valuable one. **What happened immediately before customers got angry?** Aggregate that across every conversation you hold and the answers stop being opinions: a support macro that reliably makes things worse, a billing screen that confuses everyone, a feature that produces the same complaint in five different accents, the conversations that should have been escalated four replies earlier. Per-message scoring only becomes reasonable when a judgment is cheap; that's why companies sample 1% today and why the pattern is invisible to them.

Every version of this that I've watched fail, failed by being a dashboard. Sentiment charts get looked at twice and then live in a tab nobody opens. The output has to do something: a queue of conversations for a human to rescue today, a message to the account owner while the customer is still annoyed, a weekly list of frustration triggers that lands in front of whoever owns the product area. Insight nobody acts on isn't worth a line item.

TWO THINGS TO GET RIGHT EARLY

Calibrate against outcomes. A frustration score that hasn't been checked against which accounts actually left is astrology with a nice chart. You need the churn events, joined to the conversations, before you claim to predict anything.

And be careful with the recordings. Call transcripts and support logs are some of the most sensitive data a company holds, consent rules for recording vary by jurisdiction, and "we classify every sentence your customers say" is a question you'll be asked in every security review. Have the answer written down before the first one.

THE WEDGE

One channel, one outcome.
Start with support tickets and
the accounts-at-risk queue.

WHO PAYS FIRST

A head of support or success
with a retention number and no
visibility past the sample.

THE HARD PART

Calibration against real churn,
data access, and shipping an
action rather than a chart.

07

One competitor ad is noise. Fifty thousand classified ads is a strategy map.

MARKET INTELLIGENCE · MARKETING · STRUCTURE FROM UNSTRUCTURED

Continuously collect competitors' marketing (paid social, video ads, search copy, landing pages, email campaigns, website changes) and classify every piece of it along the dimensions a marketer actually argues about. Hook. Target customer. Pain point. Promised outcome. Offer. Price. Discount. Proof. Emotional angle. Urgency. Call to action. Positioning. The objection being pre-empted.

Analysing five ads is a Tuesday afternoon. Analysing fifty thousand over eighteen months is a product, because the value isn't any single ad, it's the change: this competitor has started addressing enterprise buyers, three of them have converged on the same pain point this quarter, that one's messaging has shifted heavily toward price, annual contracts are suddenly in 40% of their creative. Those are sentences a marketing leader will forward to their CEO, and nobody can produce them by browsing an ad library.

Be clear-eyed that the model is the cheap part here. The work and the risk are both in acquisition: collecting the ads, the landing pages and the emails, continuously, legally, without depending on one platform's goodwill. Sell the weekly change digest rather than a searchable archive. Archives get cancelled; the email that says what your competitors changed while you were shipping does not.

THE WEDGE

One category with loud, high-volume advertisers. A weekly change digest, not a library.

WHO PAYS FIRST

Performance marketing leads and agencies who are already screenshotting competitor ads by hand.

THE HARD PART

Durable, defensible data collection, and a taxonomy that stays stable enough to trend against.

Every one of these is the same move: take a judgment somebody currently makes by sampling, by feel, or not at all, and make it cheap enough to make every single time.

Four more, shorter, because they're variations on the move you've now seen.

Useful less as businesses to copy than as evidence of how wide the same pattern goes. Each is a judgment somebody currently makes with rules, a sample, or a much larger model.

08

Lead scoring where the ideal customer is described in sentences, not rules.

CLASSIFICATION AT VOLUME · SALES · FUZZY SIGNALS

Most CRM scoring is arithmetic on form fields. Has email, plus ten. Has a phone number, plus ten. Fifty employees, plus twenty. It's scoring what's easy to store rather than what predicts a sale. The alternative is a few hundred fuzzy questions answered from unstructured evidence: does this company appear to have a real engineering team, does its site suggest security matters to it, does it sell into regulated industries, is there evidence of fast growth, is there anyone here who plausibly owns this problem, does the messaging suggest enterprise ambitions. Each one gets its own probability, and you combine them into a score you can explain.

The compounding part isn't the first score. It's that when a rep notices a pattern in the deals that closed, they can add that question and rescore the entire database this afternoon. That's the product: an ICP that's editable in language, not a ticket to an ops team. The hard part is distribution and behaviour: scores that don't change which accounts appear at the top of somebody's morning queue change nothing at all.

THE WEDGE

Rescoring an existing stale database on day one.
The demo is their own list, reordered.

WHO PAYS FIRST

Outbound-heavy teams burning rep hours on accounts that were never going to buy.

THE HARD PART

Living inside the CRM, and proving lift against the score they already have.

One API call in front of every piece of text your agent didn't write.

SAFETY AND ROUTING · INFRASTRUCTURE · SECURITY FILTER

Anything an AI application pulls in from outside (user input, fetched web pages, documents, emails, tool results) can carry instructions aimed at the model rather than the reader. Ignore your previous instructions. A fake system prompt buried in a page. A line in a PDF trying to talk the agent into revealing a key or calling a tool it shouldn't. Detecting that is a classification problem, and running a frontier model over every scrap of retrieved context to do the detecting is exactly the cost profile that stops teams from doing it at all.

The product is one question, asked cheaply, in a lot of places: *is this content safe to put in front of my agent?* Two honest cautions. Your adversary adapts, so this is a detection business with the maintenance obligations of one, closer to spam filtering than to a library you ship and forget. And security buyers want evidence: a benchmark, a false-positive rate, and a clear story about what happens when you're wrong. This is also the idea on the list most likely to arrive as a free platform feature, so the version worth building is broader than injection alone: policy enforcement, data-exfiltration checks, and a log somebody can audit after an incident.

THE WEDGE

A single endpoint plus a dashboard of what it caught. Easy to try, easy to rip out.

WHO PAYS FIRST

Teams whose agents read untrusted content at volume: email, web, customer uploads.

THE HARD PART

Adversarial drift, publishable accuracy numbers, and platform vendors giving it away.

Decide which model should answer before you pay a model to answer.

SAFETY AND ROUTING · INFRASTRUCTURE · AI FINOPS

Not every request needs the largest model. Classify the request first (is it simple, does it need reasoning, is it code, is there sensitive data in it, will it need tools, is there enough context to answer at all, how hard is this actually), and route accordingly. Trivial classification to something tiny, ordinary generation to something cheap, genuine reasoning to a frontier model, high-risk to a person. The router only makes sense if the router is dramatically cheaper than what it's routing between, which is the entire reason a decision model belongs in this position.

Sold as cost control, this gets meetings immediately, because most companies running AI in production have a bill that grew faster than anyone forecast. The thing to be careful about is that quality regressions from bad routing are invisible until a customer finds one. Which means

the real product is evaluation infrastructure with routing attached, because you cannot credibly move someone's traffic to a cheaper model without showing them, continuously, that the answers didn't get worse.

THE WEDGE

Shadow mode: score real traffic, show the savings you would have made, change nothing.

WHO PAYS FIRST

Whoever owns a model bill large enough to have been asked about it by finance.

THE HARD PART

Proving quality held. Without evals, you're asking for trust you haven't earned.

11

An inbox that knows a message can be twelve things at once.

CLASSIFICATION AT VOLUME · PRODUCTIVITY · OVERLAPPING LABELS

Folders are a single-label system for a multi-label problem. One message is simultaneously a customer complaint, a churn signal, an invoice question, urgent, and something legal should see. Classify along dozens of dimensions instead of sorting into three boxes, and let the user invent a new dimension whenever they need one: *find me the emails where somebody sounded interested and nobody followed up*. That question can't be answered with keywords, and with cheap classification it can be answered against the whole archive rather than the last week.

Consumer email is a famously unforgiving market, a graveyard of good clients that couldn't get paid, so the version I'd back is infrastructural. Sell the classification layer to the products that already own a queue: CRM, support desks, recruiting tools, shared team inboxes. Same engine, a buyer with a budget, and none of the burden of convincing a person to change where they read their mail.

THE WEDGE

Shared team inboxes where a missed message has a named cost. Not personal email.

WHO PAYS FIRST

Products with an inbox inside them, or teams running sales and support out of one mailbox.

THE HARD PART

Mailbox access, privacy review, and label sets that stay meaningful as they grow.

Four questions that generate more of these than any list can.

The list above is eleven answers. These are the questions that produced them, and they work better than the list, because you know an industry I don't.

I keep a folder of these. Most of them stay in the folder, and the four prompts below are how things get into it. Ask them about a business you already understand rather than about technology in general.

1. **Where are humans making thousands of tiny judgments?** Moderation, lead qualification, support triage, underwriting, quality assurance, compliance review, resume screening, categorising anything. These jobs are already decisions; they're just decisions with a salary attached.
2. **Where is somebody paying a frontier model to answer a tiny question?** If a thousand tokens of context go in and one word comes back, that call is a candidate for replacement, and the person paying for it can see their own bill.
3. **Where do people click around websites all day?** Procurement, insurance, logistics, government portals, travel, real estate, vendor research, competitive monitoring. Look for industries where the systems of record have no API and never will.
4. **Where would a probability be more useful than a verdict?** Anywhere a binary answer forces either a human review of everything or a rule that's wrong at the edges. Confidence bands let software automate the easy middle and escalate the rest, which is how this becomes a judgment layer inside existing products rather than a replacement for them.

Then write the spec before you write any code. One page, filled in honestly, kills about half of the ideas I've been excited about, usually at the line where I have to say what the customer does today and what it costs them.

COPY THIS

THE JUDGMENT

[One sentence. The single decision this product makes, over and over.]

Made today by: [a person / a rule from 2014 / a frontier model / nobody]

Volume: [how many times a day, at one customer]

THE QUESTIONS

1. [A question answerable from the context you'll actually have]

2. ...

3. ...

(If you can't write five, this is a feature. If you write forty, pick the five that would change what somebody does tomorrow.)

THE BANDS

>95% [automate: what happens with no human involved]

70–95% [review later: whose queue, and how fast]

40–70% [ask: what question gets asked, of whom]

<40% [reject or escalate: to where]

THE BASELINE

Cost today: [hours, salary, model spend, or errors shipped]

Coverage today: [what fraction gets checked at all]

THE MEASUREMENT

Labelled set: [how many examples, labelled by whom, by when]

Per-question target: [precision and recall you must hit to be usable]

Cost per decision: [and the volume at which it stops working]

THE ARTIFACT

What the customer receives: [a file, a queue, an alert, a blocked action]

Where it lands: [inbox, CRM, Slack, their existing tool]

WHAT BREAKS IT

[The site changes / the platform ships this free / accuracy isn't enough for the regulated version / the buyer has no budget line]

Teal marks what you replace. The two sections that do the real work are **the baseline** and **what breaks it**. If the baseline has no number in it, you have an interest rather than a business, and if nothing plausible goes in the last box, you haven't looked hard enough yet.

And the eleven in one place, for scanning. The right-hand column is the one to read first, because it's where the year of your life actually goes.

THE IDEA	CATEGORY	THE HARD PART
01 · Talk-to-Shop	Browser · Consumer	Variants and stock, bot defences, and an affiliate model that survives an agent-driven session.
02 · Vertical browser worker	Browser · B2B	Logins and MFA, silent breakage, and knowing you failed before the customer notices.
03 · Classified document search	Classification	Measured recall per question, and chunking that doesn't cut the meaning in half.
04 · Agent action safety layer	Safety · Infra	Sitting in the hot path, false blocks, and frameworks shipping a free version.
05 · Marketplace quality engine	Classification	Calibration you can defend, appeals, and thresholds that aren't just guesses.
06 · Churn and frustration intelligence	Customer intel	Joining scores to real churn events, data access, and shipping an action not a chart.
07 · Competitor ad intelligence	Market intel	Durable data collection, and a taxonomy stable enough to trend against for years.
08 · Intelligent lead scoring	Classification	Living inside the CRM, and proving lift over the score they already ignore.
09 · Prompt injection firewall	Safety · Infra	An adversary who adapts, publishable accuracy, and platforms giving it away.
10 · AI model router	Safety · Infra	Proving quality held after you moved somebody's traffic to a cheaper model.
11 · Communication classification	Classification	Mailbox access, privacy review, and label sets that stay meaningful as they grow.

Three ways a business like this dies, said plainly.

I'd rather you hear these from me than from an investor, and all three are survivable if you plan around them from the first week.

The first is that a decision layer is a feature until you make it a product. Every idea here can be built in an afternoon as a thin wrapper, and thin wrappers get absorbed by whoever already owns the workflow. The defensible part is never the classifier. It's the map of two hundred portals, the question library refined over two years, the labelled set nobody else has, the audit trail a regulated buyer needs, the fact that you land in the tool they already have open. Decide in week one which of those you're accumulating, and treat the model as the interchangeable part it is.

The second is that cheap only changes anything if you measure. The whole premise is doing a million judgments instead of a thousand, and a million slightly wrong judgments is worse than a thousand careful ones, because now the error is systematic and invisible. You need a labelled set, per-question accuracy, and the discipline to report it. This is unglamorous work that founders skip, and it's the exact thing an enterprise buyer will ask about in the second meeting.

The third is that browser work breaks and verification costs money. Sites redesign, add defences, and change flows without telling you, so the browser businesses carry a permanent maintenance load. And in anything regulated or financial, a decision that's right 96% of the time still needs a human for the remaining 4%, which means your margin story has a person in it. Neither is fatal. Both belong in your pricing rather than in a surprise six months in.

**The good news is that all three are execution problems.
This is a category where being boring and careful is the competitive advantage.**

One last framing, and it's the one I'd actually build on. You are not selling artificial intelligence to anybody. You are selling a judgment that used to be made by sampling, by a rule, or by a tired person at 4pm, and is now made every time, consistently, with a number attached. That's a sentence a buyer understands without believing anything about the future.

ONE MORE THING

If you're weighing one of these and want a second read.

The useful conversation is never about the model. It's about which judgment you've picked, who feels the pain, and what you're accumulating that a competitor can't copy in a weekend.

Send me the judgment your product currently makes badly, or the idea from this list you can't stop thinking about, plus a sentence on who you think pays for it. I'll tell you where I think it's strong, where I think it's a feature rather than a company, and what I'd build first. No code and nothing proprietary needed.

GET IN TOUCH

Email hi@davecto.com with the subject line "Jev Startup Ideas."

More guides like this one, for people trying to build software without fooling themselves.

Weekly, plain-language breakdowns on Instagram.

[@davecto](https://www.instagram.com/davecto)

A note on sourcing. Jev is new, and the description of it here is deliberately limited to two claims: that it is built for classification and decision-making and returns probabilities rather than prose, and that Jev-Ultrafast can use those decisions to operate a browser. Those claims come from how the model is being positioned, not from benchmarking I've done myself. I have quoted no prices, no latency figures and no accuracy numbers anywhere in this guide, on purpose: every argument here is economic, the economics change monthly, and a figure printed today would mislead you in November. Check current pricing and run your own evaluation on your own data before committing to a cost assumption, and treat any claim that a workflow "becomes affordable" as a hypothesis you test in a week rather than a fact. The ideas

themselves are not original or proprietary. Several are obvious enough that other people are already building them, which is information rather than a reason to stop. Where I've said what the hard part is, that's my judgment from watching this kind of product get built and sold, not a measured claim. I have no relationship with the makers of any model mentioned here.