

Visa Document Engine v2

Owner	Manvendra Singh Tanwar, Product
Status	Draft, for review
Written for	CTO, AI engineer, Ops lead, Founder
Version	0.3
Last updated	August 2026

Written independently, outside my work product. I own the system this describes, so the thinking is real, but volumes, unit costs and vendor terms are rounded off or left out. Nothing confidential is in here.

Summary

Our document engine works but cannot scale. Every new country costs engineering time, one confidence threshold governs every field, and we find out about consular rule changes from rejections.

v2 fixes all three. Estimated worth is \$1.0M to \$1.8M in new-market ARR over eighteen months plus roughly two engineer-years released, against about one engineer-quarter of net new build. I want approval for stages 0 and 1, which is twelve weeks and touches nothing in production.

1. Where we are

A visa application is a paperwork problem. Six to twelve documents per application, each read, checked against that country's rules, typed into a government portal, chased to a decision. Rules differ by country, visa type and nationality, and consulates change them without notice.

I built the engine that automated this and I still own it. Turnaround went from about fifteen hours to about one. Cost per application fell roughly three quarters. A human now touches under 2% of cases. It runs across thirty plus countries at roughly 35,000 to 40,000 applications a month.

It works. It will not scale to the next thirty countries, for three reasons.

2. What breaks

Every new country costs engineering time. The reader uses logic written for specific document templates, so every new country needs a developer, and so does every government form redesign. One country changed its form twice in a single quarter and we rebuilt it both times. Thirty more countries at this rate is a hiring plan, not a roadmap.

One confidence threshold covers every field. My AI engineer and I picked a single number: under about 85% sure, a human looks. That number now treats a passport expiry date and a middle initial identically. The review queue mixes dangerous cases with harmless ones in arrival order. I designed this. It was the right call to launch and it is now the main thing capping quality.

We learn about rule changes from rejections. A consulate quietly changes a requirement and nothing notices. Applications start failing, travellers contact support, support tells ops, ops tells me. Days later I have a batch of affected people and a refund conversation.

3. What this is worth

Estimates, with the assumption stated so they can be argued with. The one marked *to confirm* needs finance before I would commit to it.

New market revenue	\$1.0M to \$1.8M ARR over 18 months	30 new countries at \$35k to \$60k each in year one, benchmarked against what our current long-tail countries actually do
Engineering time released	~2 engineer-years	30 countries at 15 to 20 dev days each today, against ~2 days on configuration
Reviewer capacity released	~0.4 FTE	~800 escalations a month at ~12 minutes each, 40% of that time removed
Rejection cost avoided	<i>to confirm</i>	I do not have the all-in cost of a rejection, including refund and support handling. It is the first thing I would measure.

Cost to build. Two engineers plus me for about twelve weeks to reach stage 2, then rolling country by country. Call it one engineer-quarter of net new work, because stages 0 and 1 largely replace effort we would otherwise spend maintaining template logic.

Payback. The engineering time released covers the build inside a year on its own. The revenue is the reason to do it now rather than next year, since every quarter we wait is a quarter of long-tail countries we are not selling.

If we do nothing. Country expansion stays linear in headcount, which means the thirty-country roadmap needs two more engineers instead of two more weeks. Rejection-driven rule discovery stays our largest source of support contacts and refunds.

4. Who this is for

The agency operator on our platform, whose job is to submit forty applications and stop thinking about them. Today they cannot, because they have no idea which three will go wrong.

Behind them: the reviewer who needs the risky cases first and gets them in arrival order, the ops lead who owns turnaround targets with no early warning, and the traveller who finds out about a document problem after paying.

One sentence for what the engine should do: make paperwork invisible to the agency, and where it cannot be invisible, be specific and early about what is wrong.

5. What v2 does

1. Add a country by writing configuration, not code. Target: 80% of new countries live without touching the reader.
2. Replace the single threshold with one per field, weighted by what the field can cost us. Target: 40% less reviewer time per escalated case, rejections flat or better.
3. Catch rule changes from other signals first. Target: 70% caught before the second rejection.
4. Unit cost does not go up.

Not doing: predicting consular decisions, because we prepare and submit and they decide. Removing humans, because the goal is pointing their attention at the right cases. A chatbot, because agencies want

an empty queue, not something to talk to. Replacing v1 wholesale, because v2 goes in behind the same interface with v1 as fallback.

6. How it works

Intake -> Read -> Check -> Submit -> Track

Intake asks two questions before we spend money: is this the document it claims to be, and can it be read. A blurry passport should bounce back in five seconds. Today it fails at the end of the pipeline.

Read is the change. Today template logic runs first with a vision model as backup. Flip it: general vision model as the default, extract into fixed fields, keep template logic only where it is measurably better. Every field returns a value and a confidence, never a value alone.

Check applies country rules in plain code, not a model. Validity windows, photo sizes, balance minimums, document dependencies. If a model is deciding whether a passport has six months left, somebody built the wrong thing.

Submit is portal automation, unchanged. Government sites change constantly and no vendor keeps up.

Track polls for status.

7. Build or buy

Buy the vision model and the document classifier. That market moves faster than we can, so we call an API behind our own interface and switch when something better appears. Model prices have fallen sharply every year we have been running, and only the interface makes that a saving rather than a rewrite.

Build the rules engine, the portal automation and the test harness.

The advantage is not the model. Any competitor can call the same API tomorrow. What they cannot get is our library of strange cases: a passport variant we had never seen, an applicant with one legal name, a bank statement in an unexpected script. Every one of those already broke us once. That library plus the rules engine is what is worth owning. The model we rent.

8. The change that matters most

Stop pretending every field is equally dangerous.

Field type	Examples	If confidence is low
Dangerous	Passport number, expiry, name, DOB, nationality	Always goes to a human. Never auto-submitted.
Important	Address, employer, dates, money figures	Cross-check against the application. Escalate only on conflict.
Harmless	Middle initial, title, formatting	Fix quietly, log, move on.

One threshold forces the same tradeoff on all three rows. That is why the queue is full of noise, and it is where the 0.4 FTE in section 3 comes from.

The lesson I took from v1: **these systems live or die on the escalation logic, not the model.** We spent about 70% of development time on confidence thresholds and the handover to humans, 30% on the AI. That ratio felt wrong at the time and it was correct. Swapping models is an afternoon. Getting the handover right is the product.

9. Catching rule changes

Cheapest first: two rejections against the same country and rule in a short window raises an alert. Reactive, and still better than nothing.

Then watch pass rates per rule. If 98% of applicants used to clear a balance check and this week it is 80%, something changed on their end. That shows up before rejections do.

Last, diff the consular requirement pages weekly. Any change creates a task for the ops lead. Deliberately a task, not an automatic update, because a model reading a government website and silently changing production rules is the wrong place to hand over control.

10. Data

Passport and bank data belonging to people who agreed to get a visa and nothing else. So: no vendor who trains on it, zero retention or we walk. Redact before anything enters a test set or a debugging tool. Reviewers see the fields their case needs, not the record. Delete on a schedule once an application is final, without anyone having to remember.

11. The humans

The queue is sorted by arrival, which means it is not sorted. Sort by risk and travel date.

A reviewer should see the value, the part of the document it came from, the confidence, and the rule in question. They confirm or fix one field. They should never retype a document, which is what happens today on the worst cases.

Every correction a reviewer makes is a labelled example, and we throw all of them away. Someone fixes a misread expiry and it dies in the ticket. If corrections do not flow back into the test set, the engine cannot improve and the reviewer redoes that work next week.

Every stage also needs a defined way to fail. Model down, use template logic. Template logic down, manual with a warning to the agency. Portal failing, queue for manual. The agency must see the degraded state, because a silent slow failure costs more trust than a loud one.

12. Speed and cost

Ten minutes end to end: 5s intake, 90s to read all documents, 5s to check, 5 min to submit. Human review gets its own half hour.

Reading is the only stage where paying for more time is worth it. A slower, better model on a passport number is cheaper than a rejection.

Cost is driven by document count, image quality and model tier. Levers in order: route simple typed forms to a cheap model, reject unreadable documents before paying to read them, cache by document hash because travellers resubmit the same passport constantly, batch where the ten minutes allow.

One guardrail: cost per application is watched daily with an alert, not discovered at month end. A change that adds a point of accuracy and doubles the bill is a real decision and it should reach a person the day it happens.

13. What could go wrong

The failure that worries me is a wrong value that looks right: high confidence, passes every rule, nobody looks, application rejected. The dangerous-field rule above exists for that, and I would treat it as a hard limit rather than a target.

Then: a vendor updates their model and quality moves while we changed nothing, so versions get pinned and nothing ships without passing the test suite. Someone hides instructions inside a document, so documents are data and instruction-like text gets flagged rather than followed. A rule change slips all three signals, so rejection clustering stays as the backstop.

And the quiet one. A reader trained mostly on Western name formats does worse on single names, patronymics and long compound names, and that lands on specific passports. So accuracy is reported per country permanently, never as one blended number that hides it.

14. How I would know it is working

Two numbers on the wall: share of applications completing with nobody touching them, up. Rejections caused by our own data errors, down.

Underneath: accuracy per field split by type and country, never an average. How often an escalated case actually needed a human, target above 60%, because today most do not. Reviewer minutes per case, down 40%. Cost per application, and cost per successful submission, because those diverge when quality slips. Engineering days per new country, which is the number that decides whether the next thirty are fundable.

Two more to tell me I am fooling myself: rejection rate on auto-processed cases specifically, because you can always automate more by lowering the bar, and errors reported by agencies, because if the engine is confidently wrong they notice before our dashboards do.

15. Shipping it

Stage	What happens	Gate to move on	Time
0	Build the test set and harness. Production untouched.	We know v1's real accuracy per field, per country	3 weeks
1	v2 runs alongside v1 on live traffic, answers compared not used	v2 matches or beats v1 on dangerous fields	4 weeks
2	v2 primary on three high-volume countries, v1 fallback, per-field thresholds live	No rejection increase over a full cycle	5 weeks
3	Expand by country tier, rule-change detection on	Escalation precision above 60%	Rolling
4	v2 default, new countries by configuration	Engineering days per country at target	Rolling

Stage 1 looks like doing the work twice for no output and it is the first thing anyone will try to cut. It is also the only honest way to compare two systems on the same real traffic before anything is at stake.

What I am asking for now: approval for stages 0 and 1 only. Twelve weeks, two engineers, nothing touching production. At the end of stage 1 we will know whether v2 is actually better, measured, and the stage 2 decision can be made on evidence instead of on this document.

16. Open questions

1. What does a rejection actually cost us, all in? Needed to size the accuracy-versus-price tradeoff on dangerous fields. **[me, with finance]**
2. Should agencies configure country rules themselves? Some know a country better than we do. Also a liability surface. **[me and ops lead]**
3. What is the right review rate on auto-processed cases? Too low and problems hide for weeks, too high and the cost advantage disappears. **[me and AI engineer]**
4. Is the failure library a product on its own? A pre-submission document check sold to people who are not our customers. **[parked]**

Companion document: how the reading step gets tested, and what has to pass before a change ships.

Version history: 0.1 first draft. 0.2 added per-field thresholds after AI engineer review. 0.3 added sizing and staged ask.