E2E Networks

Real-time decisions in the payment path

How to build transaction risk scoring, routing models and dispute triage on Indian infrastructure, in a way your risk committee will sign off.

Four rules before you start

Any AI system you put in front of a payment in India has to clear the same four rules. They aren't a compliance topic and they aren't somebody else's problem. They decide what you're allowed to architect, which decides what you can ship.

Rule 1

Consent is the only door

The DPDP Act gives you two ways to process personal data. Consent, or one of nine specific uses written into the law. That's the list. There's no "legitimate interest" route the way GDPR has one, so you can't write an assessment arguing that your interest outweighs the customer's.

For an AI team it lands in one place. If you train on customer data, the consent has to cover that use. Which means you need to be able to show which data went into which model. Not describe it. Show it.

The biggest penalty in the Act, up to ₹250 crore, attaches to failing to keep data secure. Not to paperwork. To architecture, access control and who else can reach your tenancy.

The obligations and the penalties commence on 13 May 2027.

Rule 2

The data stays in India

Payment data has to be stored only in India. If you process it abroad you have to delete it there and bring it back within one business day or 24 hours, whichever is shorter. Borrower data under the digital lending rules works the same way.

Teams tend to find the awkward part late. An inference call is processing. So if your model sits outside India, every scored transaction turns into something you have to prove you cleaned up. Not once. Once per transaction, for as long as the system runs.

An endpoint inside the country removes the proof, not just the risk.

Rule 3

You validate the model, not your vendor

RBI's draft guidance on model risk covers models you built and models you bought. Two lines in it change how you design.

You validate third-party models yourself, whatever certifications the vendor holds. Their audit report isn't your validation. And a human stays in command, meaning someone who can look at what the model did, disagree with it, reverse it, and switch it off. Override, suspension and deactivation are named requirements, not good practice.

Both are cheap if you designed for them. Close to impossible to retrofit.

The guidance was issued in draft in June 2026 and the final version is awaited.

Rule 4

Check whose rules you are under

RBI regulates banks and NBFCs. IRDAI regulates insurers.

If you're an insurer, none of RBI's AI material binds you. Not the FREE-AI framework, not the model risk draft. DPDP applies to you in full, the rest doesn't. Worth checking before you borrow an architecture note from a bank.

WHERE THE FOUR RULES LAND

In five places in any stack. Jurisdiction, meaning who can be legally compelled to hand over your data. Infrastructure, meaning where the GPUs physically are. Tenancy, meaning who else can reach them. Model lifecycle, meaning whether you hold the weights and the version history. And serving, meaning whether the off switch belongs to you. Four of those are engineering choices you can revisit. The first is a fact about your provider, and it's the only one you can't change later

Five things you will be asked to show

Models don't usually get blocked for being inaccurate. They get blocked because nobody can produce the evidence. These five are what a risk committee, an auditor or a supervisor actually asks for.

A data flow page

One page. Which data, for which purpose, under which consent, sitting where. Your DPO asks first and then everyone else does. Real means it was written while the system was still small, and updated when it changed. Not reconstructed in month nine.

Version lineage

This dataset version produced these weights, which are serving on this endpoint, traceable in both directions. Model validation asks for it, and so does anyone investigating a single decision. Real means a repository you can query, not a spreadsheet somebody keeps.

An evaluation record

What you tested before you deployed, against what baseline, with what result. Your risk committee asks, under the model risk guidance. Real means the same harness every time, kept for every version, including the ones you rejected.

An override path with a name on it

How a person reviews, challenges and reverses what the model decided. Anyone who has to defend a decision to a customer will ask. Real means a named owner, a queue they actually work, and a turnaround you have written down.

A deactivation procedure you have tested

How you switch the model off, and what the business does while it's off. The model risk guidance names it and your CISO will ask too. Real means tested at least once, in production, on purpose.

THE STOP RULE

If a build can't produce all five, it doesn't reach production. That isn't a compliance problem, it's a sign-off problem, and it stops the project just as dead. Usually around month eight, after the model has already shown it works.

Where the line sits. Our certifications cover the infrastructure and the platform. Your application's compliance posture stays yours. We can give you the architecture, the evidence and the contract terms to close the gap. We can't close it for you, and neither can anyone else.

The smallest agent you will ever ship

Everyone at this event is talking about systems that sense, decide and act in real time, inside strong governance. A transaction risk model does exactly that, thousands of times a second. Most teams don't think of it as an agent.

They should. It senses a transaction, it decides, and then it acts. It declines, or steps up, or lets through. That third verb changes everything, because the moment a model acts rather than advises, every requirement in the previous section attaches to it.

There's an upside to that. Get this one right and the harder agents are the same problem at lower speed.

Three things you are probably building

Transaction risk scoring. A fraud and step-up decision on every authorisation. Sense, decide, act.

Routing and success rate models. Issuer route, retry strategy, timing. The easiest business case in payments to sign off, and the hardest to run cheaply, because it needs a decision on every attempt.

Dispute and chargeback triage. Batch, document heavy, tolerant of latency. This one behaves completely differently from the other two, and that matters more than it sounds. It can run in the hours your authorisation traffic leaves idle, which is the difference between paying for peak capacity and using it. It comes back in decision four.

Three things kill it, in this order

The tail, not the average. You know your latency budget. It isn't an average, it's a promise you have to keep almost every time, because the tail is what the customer and the issuer see. A shared multi-tenant endpoint has a tail you don't control and can't debug.

The localisation proof. Rule 2 above. If the model sits abroad, every scored transaction becomes a delete and repatriate you have to evidence. Once per transaction, forever.

Cost per decision at full volume. This one arrives last and hurts most. The pilot ran on sampled traffic and the bill was trivial. At authorisation volume the same architecture becomes a line item somebody notices, usually after the model has already proved it works. Which is the worst possible moment to be told no.

WHAT YOUR BOARD ALREADY THINKS

RBI's Financial Stability Report of June 2026 surveyed 33 scheduled commercial banks and 10 upper-layer NBFCs on cyber risk. **95% put AI-enabled threats in their top three** for the year ahead, ahead of third-party risk at 70% and ransomware at 28%. On their own readiness for those threats, 45% said developing, 38% said intermediate, 5% said mature, and **none said advanced.** Nobody is asking you to justify why. They're asking you to ship something defensible.

Nine decisions

Most of the architecture argument in payments AI is settled before you start, by the four rules above. Here is where each one lands.

THE DECISION	MOST TEAMS REACH FOR	WHAT SURVIVES THE FOUR RULES
Model class and size	The largest general model available, through an API	A mid-sized open-weight model, fine-tuned on your own transaction history. Narrow task, your labels, your lineage. Also the only version whose cost per decision works at volume.
Where it runs	A managed endpoint, wherever the provider puts it	A dedicated endpoint on your own cluster, inside your VPC, in India. Rule 2, and the only way to own the tail.
Serving stack	Whatever the provider offers	vLLM or SGLang on dedicated GPUs, with MIG partitioning where the model does not need a whole one. Continuous batching is what buys throughput without losing latency.
Sync, async or batch	Everything synchronous	Scoring synchronous. Dispute triage as a scheduled job on the same cluster, overnight. Batch work fills the troughs the authorisation peak leaves. Biggest single lever on blended cost, and only available when the capacity is yours.
Context retrieval	Merchant and device history stuffed into the prompt	A retrieval step against your own store, fetched at decision time. Prompt size is latency and cost. A retrieval boundary is also somewhere you can log what the model actually saw.
Where the human sits	Nowhere. It is real-time.	Not in the path, in review. A named owner working a queue of flagged and reversed decisions, with a written turnaround. Human in command does not mean a human on every transaction.
What happens when the model is down	Nobody has decided, so the answer is that payments stop	The deterministic rules engine stays in the path. Suspend the model and the rules still authorise. Your off switch and your availability answer in one choice. The system degrades instead of stopping.
How you will see what it is doing	Model metrics in a notebook, checked when something looks wrong	Request-level logs, latency percentiles and the decision distribution, shipped into your own monitoring from day one. You need the logs for the audit trail anyway. Build them once and use them twice.
How you will be billed	Variable per-token pricing, reconciled monthly	Dedicated capacity at a published hourly rate, billed at a fixed amount you can model in advance. You cannot manage a cost per decision you cannot predict.

IF YOU ONLY TAKE ONE ROW, TAKE THE SEVENTH

Any AI component in the authorisation path will be asked what happens when it fails, usually by someone who can stop the project. Keeping the rules engine live means the honest answer is that approval rates move and nothing stops. It also gives you the off switch your model risk policy needs, at no extra cost.

Five things break, and only one is the model

These are the questions you will be asked around month four.
The point of this section is that you can answer them in month one.

Your labels arrive months late

Ground truth in fraud comes back as chargebacks, weeks or months after the transaction. So your training set is always stale, and your offline evaluation is always more optimistic than production. Hold out by time, not at random. Measure on a window old enough for labels to have matured. And expect your first production numbers to be worse than your test numbers. That gap is the shape of the problem, not a fault in your model.

The adversary adapts, and nobody else's does

Document models drift because the world changes. Fraud models drift because someone is deliberately probing them. Watching input distributions isn't enough. Watch the decision distribution and the approval rate, because a fraudster who has found a gap shows up there first. And have a retrain path you have actually run, not one you have written down.

The tail moves at peak

Your P99 on a Tuesday afternoon tells you almost nothing about your P99 on a sale day. Load test at the peak you actually have. Autoscale on concurrency or requests per second. Not on GPU utilisation, which lags, and by the time it reacts the queue has already formed.

Cost arrives after the business case

Cost per decision isn't the hourly rate. It is the rate divided by the decisions per hour you actually achieve, plus whatever you pay to move data. The denominator dominates, so utilisation moves the number further than price does. And data movement is the term nobody models, because it's trivial in a pilot and linear in production. Work it out before you scale.

The base model changes and nobody re-runs the evaluation

You upgrade to a newer open-weight release because it benchmarks better, and it does, on general benchmarks. On your task it may well not. Worse, it might behave differently on exactly the edge cases your rules were tuned around. Re-run the same harness, compare against the version in production, and keep both results. This is usually the moment a team discovers that its evaluation harness only ever existed as a notebook somebody ran once.

The same five break every agent you build after this one. A vernacular collections agent has the same label delay problem with different labels. A dispute agent has the same tail problem at a tenth of the speed. What makes a transaction risk model the right place to learn all of it is frequency: at authorisation volume you find out you were wrong in days rather than quarters.

Running it

Everything above runs on ordinary platform capability. No custom work, no professional services engagement, and nothing waiting on a roadmap.

What you need from the platform

- **Dedicated GPU nodes** at a published hourly rate, billed at a fixed amount, so you know the cost before you commit instead of reconciling it after.
- **Endpoints inside your own network,** attached to your VPC and behind your security groups, so the scoring call never leaves your perimeter or the country.
- **Open-weight models you hold,** versioned alongside the datasets that produced them.
- **Fine-tuning on your own data,** with a way to score the result against your baseline before you deploy it, and to keep that result.
- **Scheduled jobs on the same cluster,** so batch work fills capacity you have already paid for.
- **Metrics and logs you can pull into your own monitoring,** not a dashboard somebody has to log in and read.
- **A REST API for all of it,** so none of it is a console workflow.
- **Support from engineers who work on the platform,** reachable directly, not through a queue that answers with a link to the documentation.

On billing

Rates are published, and dedicated capacity bills at a fixed rate, not a variable one. That means your cost per decision is something you can calculate in advance and defend in a business case, instead of something you discover in the monthly invoice.

It matters more in payments than almost anywhere else. The volume is relentless, and a small per-call cost compounds into a large annual one. A model that clears compliance and clears validation can still fail to ship because nobody could predict what it would cost at full traffic.

On observability

You'll end up building the same telemetry twice if you aren't careful. The logging your audit trail needs and the logging your engineers need overlap almost completely: which model version served this decision, what it saw, what it returned, how long it took. Build it once, at the endpoint, and ship it into whatever you already use.

Two signals matter more in payments than the usual ones. Latency percentiles, not averages, because the tail is the promise. And the decision distribution over time, because a shift there is your earliest warning that either the traffic or the adversary has changed.

Thirty, sixty, ninety

Days 1 to 30

Measure, don't demo. Stand up a node, load an open-weight candidate, and get two numbers on your own data shape: decisions per GPU-hour achieved, and P99 at your peak concurrency. While that's running, get compliance to write the one-page data flow. Exit: two measured numbers. Not a demo.

Days 31 to 60

Move onto a dedicated cluster attached to your VPC. Fine-tune on your own labelled book and keep the evaluation output, because that's the start of your file. Version the dataset and the weights together. Then pull the rules-only fallback in production, once, deliberately, and watch what happens. Exit: a model that beats your baseline, and a fallback you have actually tested.

Days 61 to 90

Serve from the dedicated endpoint. Turn on logging that satisfies an audit trail, not a dashboard. Write the model into your inventory with its version, its data version, its owner and its off switch. Take it to your risk committee with the evaluation record in hand. Exit: one model live, one file complete, and a second use case that costs almost nothing to add.

Two numbers this guide does not print

Cost per decision and achieved throughput would make it complete. Both depend on your model, your traffic shape and your utilisation, so they are yours to measure and not ours to assert. The first thirty days above exist to get you both, on your own data, before you commit to anything.

COME AND TALK TO US

Bring your volume and your latency budget. Those two numbers, plus your compliance constraint, are enough to sketch this architecture properly. We will do it on a whiteboard in about fifteen minutes.
Booth no. A2, Global Fintech Fest. Jio World Centre, Mumbai.

E2E Networks

e2enetworks.com