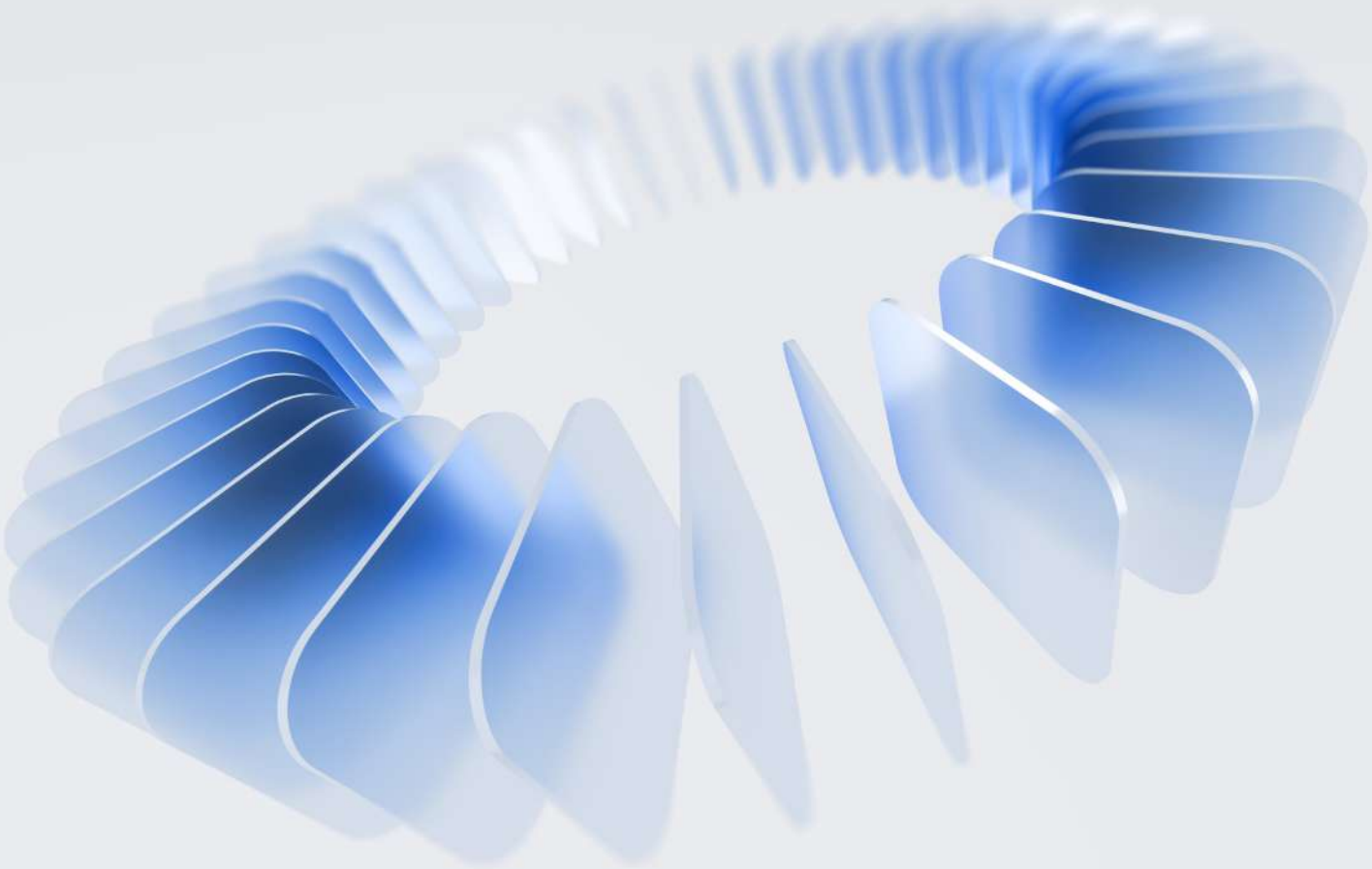
E2E Networks



Decisions you have to explain

How to build underwriting decisioning, document intelligence and vernacular collections 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 credit decision in India has to clear the same four rules. They aren't a compliance topic and they aren'tsomebody 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 whichdata 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RBI's draft guidance on model risk covers models you built and models you bought.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.

Your book is the only thing nobody else has

Every lender in India can buy the same bureau data, the same model weights and the same GPUs. What none of them can buy is your repayment history, your collections outcomes and your rejected applications. That's the asset. The model is just how you use it.

Which is why lending is the one place where holding your own weights stops being a preference. You can't produce a custody proof for a model you don't hold, and almost every question a supervisor asks about a credit model is a question about custody.

Four things you are probably building

Underwriting decisioning. Models supporting or making the credit decision. The highest scrutiny and the highest value in the segment, and the one people build last for good reason.

Document intelligence. Extraction and reconciliation over bank statements, ITR filings and GST returns. Unglamorous, enormous in aggregate, and the fastest route to a defensible win because it helps a human decide instead of deciding for them.

Vernacular collections agents. Early-stage conversations across Indian languages. High volume, tightly scripted, and the one that needs the hardest escalation path.

Early warning and delinquency models. Portfolio level, batch, and the least contentious of the four, because no single adverse decision hangs on one inference.

ONE NOTE ON COLLECTIONS THAT HAS NOTHING TO DO WITH AI

An automated dialler sits under RBI's fair practices code and its recovery agent norms, and under TRAI's rules for automated commercial calls. Both bind an agent exactly as they bind a human caller. Neither is satisfied by anything in your architecture, and this is usually the first question a lending compliance head asks about the use case.

Three things kill it, in this order

The documents in production aren't the documents you tested on. You built on clean PDFs from three banks. What arrives is phone photographs, regional formats, stamped and annotated pages, and statements where the balance column moves halfway down. This one bites first, in the step from pilot to real volume.

The consent doesn't reach the training use. Rule 1 above. If the consent taken at origination doesn't cover model training, your corpus is a problem, and you find out at the second compliance review rather than the first.

You can't explain a decline. This one arrives last and kills hardest. An adverse credit decision needs a reason a person can defend and a route for the customer to contest it. "The model said so" is not a reason, and a risk committee will not accept it however good your accuracy is.

WHAT THE SUPERVISORY DATA SAYS

RBI's own survey of 612 regulated entities found only **20.8% using or developing AI at all.** Among NBFCs it was 27%. Of the AI applications catalogued across those entities, **credit underwriting was one of the largest clusters at 13.7%,** so plenty of people are trying. The number that matters for this guide is different. Of the entities that had actually built something, only about **15% used model interpretation tools.** Five out of six teams with a model in hand cannot explain what it did.

Nine decisions

Most of the architecture argument in lending 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 book. Your labels are the asset. And you cannot produce lineage, or an independent validation, for weights you do not hold.
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. Borrower data has its own storage rule, and it is stricter than most teams assume.
Serving stack	Retrieval for everything, because it is easier to change	Fine-tune the stable thing, retrieve the volatile thing. Document layouts and extraction are stable. Credit policy and product rules are not. Policy changes weekly. If you bake it into weights you retrain every time the product team moves a threshold.
Sync, async or batch	Send the PDF straight to a general model	A layout and OCR pass first, then a fine-tuned extraction model, with a confidence score per field. General models fail quietly on bad scans. Per-field confidence is what lets you route the doubtful ones to a person.
Context retrieval	A reviewer at the end, looking at everything	A confidence threshold that routes to a review queue, with the threshold set as a business dial and not a default. This is your override path, and it is also what makes your accuracy claim auditable.
Where the human sits	Nothing, or a generic decline	Reason codes built from the features the model actually used, mapped to language a grievance officer can defend. An adverse decision needs a reason and a route to contest it. Retrofitting reason codes onto a finished model is close to impossible.
What happens when the model is down	Applications queue until somebody notices	Fall back to the existing policy rules, mark those applications, and re-score them later. Your off switch and your availability answer in one choice. Underwriting slows down instead of stopping.
How you will see what it is doing	An accuracy dashboard, checked weekly	Per-field confidence distributions, approval rate by segment, and the rate at which work lands in the review queue. Your earliest drift signal is the review queue growing, not accuracy falling. Accuracy moves last.
How you will be billed	Variable per-token pricing, reconciled monthly	Dedicated capacity at a published hourly rate, with the batch work scheduled into the troughs. Document volume is bursty. Month end and campaign spikes are exactly when you least want a variable bill.

IF YOU ONLY TAKE ONE ROW, TAKE THE SIXTH

Reason codes are the thing teams leave until last and then cannot add. If the model was designed to output a score and nothing else, there is no honest way to produce the explanation afterwards, and every workaround a vendor offers you is a second model guessing at the first one. Decide what a decline will say on the day you decide what the model will be.

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.

Reject inference

You only ever see how the loans you approved performed. The ones you declined have no outcome, so your training data is censored, and your model gets confidently worse exactly at the margin where the decisions are hard. There are established techniques for this and none of them are free. At minimum, keep a small random-approval holdout if your risk appetite allows it, and be honest in your validation file about what the model has never seen.

Your policy changed and the model didn't

Credit policy moves faster than models do. A product team adjusts a cut-off, a new state gets added, a segment gets paused. If those rules live inside the weights, every policy change becomes a retrain. Keep policy in a layer you can edit on a Tuesday, and keep the model doing the thing that doesn't change.

The review queue quietly becomes the product

Set your confidence threshold conservatively and humans end up doing most of the work. Nobody notices, because the accuracy numbers look excellent. Then someone works out the cost per application and it's worse than it was before automation. Watch the share of applications going to review, and treat a rising share as a defect rather than as caution.

Fairness gets tested once

Bias and fairness get evaluated at deployment, and then the book shifts underneath and nobody looks again. When a supervisor asks whether the model has an adverse impact on a group, the question is retrospective, and you can only answer it if you kept the evaluations. Re-run the same tests on a schedule and keep every result, including the ones you did not like.

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 documents it may well not. 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.

Notice how few of these are model problems. Four of the five are about what surrounds the model: the data you never collected, the policy layer you did not separate, the threshold nobody revisited, and the tests nobody repeated. That is the general shape of lending AI, and it is why the architecture matters more than the model choice.

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 application is something you can calculate in advance and defend in a business case, instead of something you discover in the monthly invoice.

Lending volume is lumpy in a way payments volume isn't. Month end, festive campaigns, a new partnership going live. Variable pricing puts your largest bills in the same weeks as your largest operational load, and dedicated capacity with scheduled batch work does the opposite.

On observability

The logging your engineers want and the logging your audit trail needs are almost the same logging. Which model version handled this application, what it extracted, what confidence it gave each field, what it returned, how long it took. Build it once, at the endpoint, and ship it into whatever you already use.

Three signals matter more in lending than the usual ones. The share of applications routed to human review, because that is where cost hides. Per-field extraction confidence, because a drop there precedes an accuracy drop by weeks. And approval rate by segment over time, which is the number a supervisor will ask you about first.

Thirty, sixty, ninety

Days 1 to 30

Pick the narrowest document type you have most of. Measure extraction accuracy per field on a held-out set, and pages processed per GPU-hour. While that's running, get compliance to confirm in writing whether the consent taken at origination reaches model training. Exit: a per-field accuracy number, a throughput number, and a written answer on consent.

Days 31 to 60

Fine-tune on your own labelled set and keep the evaluation output, because that's the start of your file. Version the data and the weights together. Build the confidence threshold and the review queue, and set the threshold deliberately. Exit: a model that beats your baseline, and a review queue with a number on it.

Days 61 to 90

Ship the pipeline behind the human queue. Turn on logging that satisfies an audit trail, not a dashboard. Write reason codes for anything that touches a credit decision. Take it to your risk committee with the evaluation record in hand. Exit: one pipeline live, one file complete, and a second document type that costs almost nothing to add.

Two numbers this guide does not print

Cost per application and achieved throughput would make it complete. Both depend on your documents, your model 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