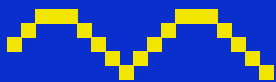


SIX

Tracks

Six domains worth building in. Pick one and send us an idea. The problem statements come later, on the beach.



-
- | | | |
|----|-----------------------------|---|
| 01 | Agents in Production | Software that does the job, and what keeps it in bounds |
| 02 | The Toolchain | Everything you build with, and what it quietly costs |
| 03 | The Trust Layer | Proving a thing is what it claims to be |
| 04 | The Rails | Rails a billion people already stand on |
| 05 | The Edge | Intelligence that runs where the thing actually is |
| 06 | Earth Systems | Water, land, weather, and everything living off them |
-

AGENTS IN PRODUCTION



Software that does the job, not software that talks about it. And everything that has to exist before anyone lets it near real work.

An agent is a system that takes an action on its own. Filing the claim. Reconciling the invoice. Triaging the ticket. Running the deploy. The demo version of this has been easy for two years.

The production version is not. The moment an agent touches real money, real records or a real customer, somebody has to decide what it may do alone, what it may

only prepare, what always waits for a human, and how anyone proves afterwards what it did.

Build here if your idea is an agent doing a real job, or the machinery around one: tool use, permissions and approvals, evaluation, memory, audit trails, or several agents coordinating.

IN SCOPE

TOOL USE

PERMISSIONS AND APPROVALS

EVALUATION

MEMORY AND STATE

AUDIT AND TRACEABILITY

MULTI-AGENT ORCHESTRATION

BACK-OFFICE AUTOMATION

NOT THIS TRACK

A chat window wrapped around an API.

WHY NOW

Four things that make this hard, and none of them are about model capability.

01

Evaluation is the bottleneck, not capability. A team can build a working agent in a week, then spend six months unable to answer whether version two is better than version one.

02

Permissions were designed for humans. Every auth system in production assumes one identity, one session, one intent. OAuth scopes cannot express “may refund up to 500 rupees without asking”.

03

Failure is silent. A crashed service pages somebody. An agent quietly doing the wrong thing for three weeks does not, and the first signal is a customer.

04

The economics push the human out. A review step often costs more than the agent run it is reviewing, so the pressure is to delete it exactly where it matters most.

WHAT GETS AN IDEA SHORTLISTED

01

The job

Which task, done by which role, how many times a day, and how long it takes them now.

02

The engineering

Reliable tool use, state across long runs, cost per task, or verifying the output. Say which one is your hard part.

03

The line you drew

What the agent does alone, what it only prepares, and what always waits for a person.

04

Why it gets used

Who hands this work over, what they stop paying for, and what they need to see before they trust it.

THE TOOLCHAIN

02

The layer every product stands on. The editor, the pipeline, the deploy target, the gateway, and the thing that tells you it broke.

Nobody looks at their tooling until it falls over, and then it is the only thing anybody looks at. This layer decides how fast a team ships, what it costs them to stay up, and how long a new person takes to become useful.

It got harder in the last two years. A real share of code is now written by a machine, almost no team can tell you which share,

and the packages underneath have turned into an attack surface.

Build here if your idea is a tool for people who build things: developer experience, build and deploy, observability and debugging, dependency safety, model gateways and cost control, or infrastructure a small team cannot afford to run properly.

IN SCOPE

DEVELOPER EXPERIENCE

BUILD AND DEPLOY

OBSERVABILITY

DEBUGGING

DEPENDENCY SAFETY

MODEL GATEWAYS AND COST

SELF-HOSTED INFRASTRUCTURE

NOT THIS TRACK

A project management app for developers.

WHY NOW

Four problems in this layer that nothing in the standard pipeline currently solves.

-
- 01 **Nobody can tell you which lines a model wrote.** Code review assumes an author who can explain intent. A growing share of merged code has no such author, and nothing records which share.
 - 02 **The dependency graph outgrew anyone's ability to audit it.** One install pulls in hundreds of packages from maintainers you will never meet, most of them unpaid, any of them one stolen token away from your build.
 - 03 **Observability now costs more than compute** for a lot of teams, so they turn tracing off. The systems under the most cost pressure end up the least instrumented.
 - 04 **Every AI feature added a gateway and nobody owns the bill.** Per-token spend lands on a card statement with no attribution to a team, a feature or a user.
-

WHAT GETS AN IDEA SHORTLISTED

-
- | | |
|--|--|
| <p>01</p> <h3>What exists</h3> <p>Name the tool people use today and the exact point where it fails the person using it.</p> | <p>02</p> <h3>The engineering</h3> <p>Why this is a build and not a config file. Latency, scale, correctness, or a protocol nobody has implemented well.</p> |
| <p>03</p> <h3>Who the developer is</h3> <p>A student on a laptop, a two-person team, a college running its own hardware. Not "developers".</p> | <p>04</p> <h3>Why they switch</h3> <p>Who moves off their current tool for this, what it saves them per week, and what it costs to run.</p> |
-

THE TRUST LAYER

03

Everything that can now be faked, and the work of proving that a thing is what it claims to be.

A voice. A video call. A software update. A model's output. A record of a payment. Every one of them can be manufactured now, cheaply, by anybody, at scale.

There is no single defence. It is detection where detection works, provenance where a thing can be proved, and sensible limits everywhere else. Most of it has to work on

people who are not technical and are being rushed.

Build here across threat detection, prompt injection and agent hijacking, deepfakes and content provenance, fraud and scam defence, identity for people and for machines, supply chain integrity, and applied cryptography.

IN SCOPE

THREAT DETECTION

PROMPT INJECTION

DEEPPAKES AND PROVENANCE

FRAUD AND SCAM DEFENCE

IDENTITY

SUPPLY CHAIN INTEGRITY

APPLIED CRYPTOGRAPHY

NOT THIS TRACK

A password manager with a new interface.

WHY NOW

Four reasons the defence is structurally behind, not merely underfunded.

01

Detection loses by design. A detector trained on this year's generators degrades against next year's. The defender has to retrain. The attacker only has to regenerate.

02

The instruction and the data arrive through the same pipe. No model reliably separates what the user asked from what the document said, and prompting cannot fix something architectural.

03

Provenance only works if the whole chain adopts it. A signature applied at capture survives until the first screenshot, and most platforms that matter do not display it anyway.

04

The expensive error is the false positive. Lock a real person out of their own money and you have caused the harm you were built to prevent.

WHAT GETS AN IDEA SHORTLISTED

01

The attacker

Who they are and what they are actually after. Not "hackers". And what is worth taking here.

02

The engineering

The detection or verification problem at the centre of it, and why a confidence score on a model does not settle it.

03

The cost of being wrong

Both ways. Missing a real attack, and accusing a normal user of being one.

04

Where it sits

Whose system this runs inside, who pays for it, and why they would let it near their users.

THE RAILS



Public infrastructure a billion people already stand on, and the software almost nobody has built on top of it.

India built something no other country has. UPI moves money between strangers for free. Account Aggregator moves financial records with consent. DigiLocker holds documents, ONDC unbundles commerce, ABDM does health, Bhashini does language.

Hundreds of millions transact on these every day, and the people furthest from a

keyboard have seen the least of it. The rails exist and they are open. The last mile, in a language somebody speaks, at a price they can pay, is mostly empty.

Build here in finance and credit, health, learning, civic services and entitlements, commerce, language and accessibility, and the informal work most software has never bothered with.

IN SCOPE

CREDIT AND FINANCE

HEALTH RECORDS

LEARNING

CIVIC SERVICES

COMMERCE

LANGUAGE AND ACCESSIBILITY

INFORMAL WORK

NOT THIS TRACK

Another UPI payment app.

WHY NOW

Four reasons the rails are open and the software still is not there.

01

Consent is built, trust is not. Account Aggregator can move a person's financial history in seconds. Nobody has made that consent legible to the person being asked to grant it.

02

The data exists and nobody will underwrite on it. Three years of UPI receipts is a cashflow record, and there is still no accepted way to turn transaction flow into a credit signal that survives an audit.

03

The last mile is a person, not an app. Most rails software that works here is operated by an agent, a shopkeeper or a cooperative on somebody's behalf. Almost none of it is designed for that.

04

Language is being treated as translation. Bhashini will give you Malayalam text. It will not give you a form that somebody who has never filled a form can complete.

WHAT GETS AN IDEA SHORTLISTED

01

One named person

Not "farmers". A fish vendor in Chavakkad who sells on credit and settles on Sundays.

02

The engineering

Which rail you need, what it actually returns, and the work left over once it has returned it.

03

The step that changes

The moment in their day that is different, and what that moment costs them today.

04

Who carries it

A bank, a cooperative, a panchayat, an agent. Somebody has to put this in front of that person.

THE EDGE

05

Computing that happens where the thing actually is. On the device, in the field, on the machine, with no signal.

For twenty years, connected hardware meant a sensor posting readings to a dashboard nobody opened twice. That era is finished. Models run on the device now. Robots act without asking a server first.

The work moved from collecting data to deciding on it, and the constraints moved with it. Power, memory, the cost of the board, and what happens on the day the

network is gone. Most places where this matters have no reliable network at all.

Build here with on-device models, robotics and drones, vision at the edge, sensor networks, industrial and agricultural monitoring, assistive and medical devices, low-power and solar hardware, and Indic speech and vision.

IN SCOPE

ON-DEVICE MODELS

ROBOTICS AND DRONES

VISION AT THE EDGE

SENSOR NETWORKS

INDUSTRIAL AND AGRI MONITORING

ASSISTIVE DEVICES

LOW-POWER HARDWARE

INDIC SPEECH

NOT THIS TRACK

A sensor that posts readings to a dashboard.

WHY NOW

Four things that break between a working demo and a device in the field.

01

The model fits and the board still does not. Quantised weights sit in flash easily, and then the thing dies on activation memory, or on thermals, on the hardware somebody actually bought.

02

Nobody designs for day two hundred. The interesting question is not what it does when it works. It is what it does once the sensor has drifted, the card is full and nobody has visited the site in months.

03

Robots handle the expected case. Picking a known object out of a known bin is solved. A spill, a person where no person should be, a dropped item, none of that is.

04

The gap is labelled data, not architectures. Indic speech and regional vision fail on the long tail because nobody collected it, not because the models are missing.

WHAT GETS AN IDEA SHORTLISTED

01

The physical place

The machine, the field, the boat, the room.
Where exactly this sits, and who is standing near it.

02

The engineering

What runs on the device and what needs a server, inside what power, memory and cost budget.

03

The decision it drives

Not what it measures. A reading nobody acts on is a dashboard.

04

Who runs it

Who buys the board, who mounts it, and who notices on the day it stops working.

EARTH SYSTEMS

06

Water, land, weather, food, and everything living off them. The system underneath every other system, and the only one nobody designed.

Every platform in the other five tracks runs on top of this one. It floods, it erodes, it dries out, and it does not wait for a release cycle.

Kerala is an unusually good place to work on it. A coast that is visibly moving, rice grown below sea level, a monsoon that arrives twice a year, and a generation of fishermen and farmers who read all of it

better than any model does and have never once been asked.

Build here on fisheries and the sea, floods and landslides, rivers and groundwater, coastal erosion, soil and food supply, waste and energy, forests and air, and the early warning that is supposed to reach a person before any of it does.

IN SCOPE

FISHERIES AND THE SEA

FLOODS AND LANDSLIDES

RIVERS AND GROUNDWATER

COASTAL EROSION

SOIL AND FOOD SUPPLY

WASTE AND ENERGY

EARLY WARNING

NOT THIS TRACK

A dashboard of climate statistics.

WHY NOW

Four reasons the data already exists and the decision still does not.

01

The warning existed. The delivery did not. In most Indian disaster post-mortems the forecast was broadly right, and the failure sat between the bulletin and the person.

02

Prediction is not the hard part, resolution is. National models forecast at a scale far coarser than the hillside or the ward that actually floods, and downscaling without local ground truth is guesswork.

03

Nobody is paid to maintain a sensor. These projects do not fail at deployment. They fail eighteen months later, when it turns out nobody owned calibration.

04

The best dataset in a fishing village is in people's heads. Decades of reading current, wind and fish movement, none of it in any form a model can use.

WHAT GETS AN IDEA SHORTLISTED

01

The person and the decision

Who lives with this, and what they must decide right now without enough information.

02

The engineering

Where the data comes from, what is missing from it, and the sensing or modelling problem that follows.

03

What already exists

Something official already measures this. Say what it is and why it has not solved the problem.

04

Who it reaches

The channel that puts this in front of that person in time. A system nobody sees is not a warning.

THE BAR

We read every application. This is what separates the ones that get in.



01 A person, not a category

"Farmers" is not a user. "Paddy farmers in Kuttanad who lease their land" is. If we cannot picture one specific human being, the idea has not been thought about yet.

02 Evidence, not assumption

How do you know this problem is real. A conversation you had, a number you found, something you watched happen. An idea that exists because it sounded good reads that way.

03 Real technical depth

Name the hard part and be honest about it. We read for what is difficult to build, not for how it is pitched. A CRUD app with a chat box on it is not a hard part.

04 You know what exists

Name the thing that already tries to solve this and say where it falls short. "Nothing like this exists" almost always means nobody checked.

05 A reason it gets used

Who it is for, why they would come back after the demo, what it costs to run, and the one number that would tell you it worked.

06 Buildable in a weekend

Ambition is good, fantasy is not. We have to believe four people could have a working version by Sunday morning. Not a research lab, not a year, not data nobody will give you.