CLARISTA WHITE PAPER · NOW WHAT SERIES · FINANCIAL SERVICES

You Built the App. Now Comes the Hard Part

Firm data, authentication, sharing, and governance for AI-built apps. Anyone can build a useful app in an afternoon now. Almost none of them reach a second user. Here is what stops them, and how to get past it.

Suvrat, CEO of Clarista · July 30, 2026 · 16 min read

Building the app is the easy part now. The hard part comes after: connecting it to your firm's data, deploying it so colleagues can use it, adding sign-in, and governing it. That is where most AI-built apps quietly die.

Someone with no engineering background sits down with Claude Code or Codex and builds a working app in a couple of hours. It reads their internal numbers, pulls in some context from the web, and produces a clean analysis. They demo it. Everyone is impressed. Then nothing happens. The app never reaches a second person. The work that would have turned it into real software was invisible to the person who built it, and that work is what this paper is about.

KEY TAKEAWAYS

- The fun part of AI, getting an output, is easy now. The parts that make an app **shippable** are the hard ones: live data, deployment, sign-in, logging, audit, and security scans.

- The **data problem looks different at every firm size**. Big firms fight internal silos and change control. Small advisory teams fight vendors who charge more for API access than the tool itself costs.
- To **share the app**, you go back to IT and wait weeks, even though building it took an afternoon.
- Both problems have the same kind of fix. **Put your data in a low-cost lake you own and serve it through governed MCP and API endpoints.** Then **deploy through a one-click pipeline that IT sets up once.**
- The **soft problems** matter too. Developers worry about their role, and leaders worry about hundreds of unmanaged apps. The answer to sprawl is not to block it. It is to make every app land in the same governed place.

This plays out at regulated firms every week now. More than four million people already build software just by describing what they want, and the tools have gone from novelty to daily habit. In financial services, almost none of what they build makes it to production. The app itself is usually fine. What stops it is everything that has to be in place around the app before the firm can let a second person use it. The builder sees the finished stage. They never see the backstage, and the backstage is where the real work happens.

01 · THE FRONT OF THE STAGE

What can you actually build with Claude Code or Codex today?

Here is a case you have probably seen. A portfolio analyst, an advisor, or someone in finance opens a coding assistant and builds a small app to analyse financial data. One tab takes a spreadsheet upload of internal numbers. Another pulls context from the web: a rate, a filing, a comparable. If they are a bit more technical, a third tab calls an outside API for extra signal. It works, and it is useful.

Something that used to need a project, a budget, and a spot in a queue now takes an afternoon and nobody's sign-off.

This is what vibe coding promised, and the promise is real. The front of the stage has changed. The mistake is assuming the rest of the show changed with it.

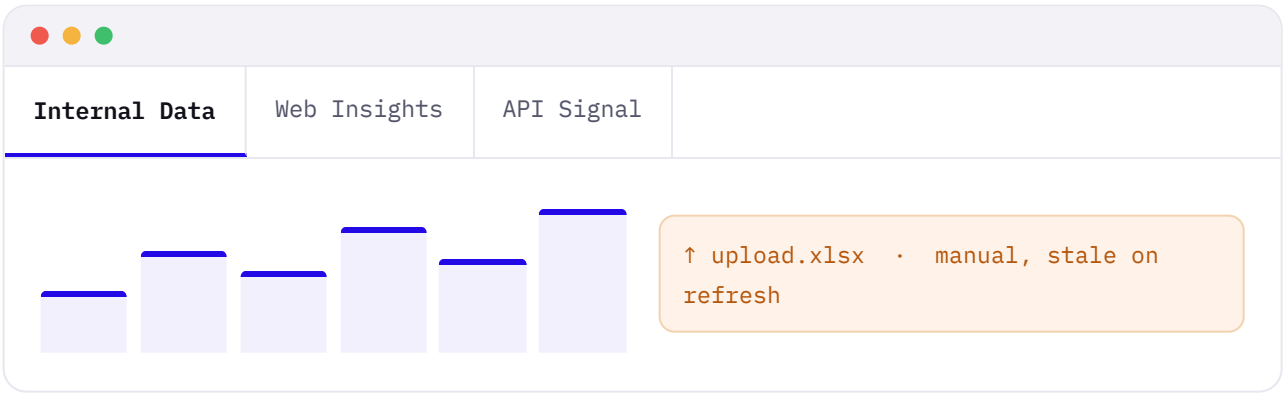

Figure 1. The finished app looks complete. The manual spreadsheet upload is the first sign that it is not.

For the wider context on how firms got here, see our primers on what vibe coding is and the best vibe coding tools of 2026.

> TAKEAWAY
>
> The demo is real. That is what makes what happens next so frustrating.

02 · THE BACKSTAGE

Why does an app that works never reach your colleagues?

Think about a Broadway show. You watch the performers, the songs, the lights. But the show only happens because of the crew backstage, and there are usually more of them than there are performers. They build the set, run the rigging, handle

the sound, and keep everyone safe. You never see any of it. That is the point. The part you enjoy is the easy part. The hard part is everything it takes to put it on.

AI-built software works the same way. The output is front of stage, and that part is easy now. Everything else is backstage: getting real data in, deploying the app, adding sign-in, logging what it does, keeping an audit trail, storing secrets safely, and passing security scans. The business team that built the app does not understand this work and has no interest in doing it. So the builder gives up on sharing it. They keep the app to themselves, re-upload the spreadsheet by hand when they need fresh numbers, and get a fraction of the value out of it. A good idea sits there and slowly dies.

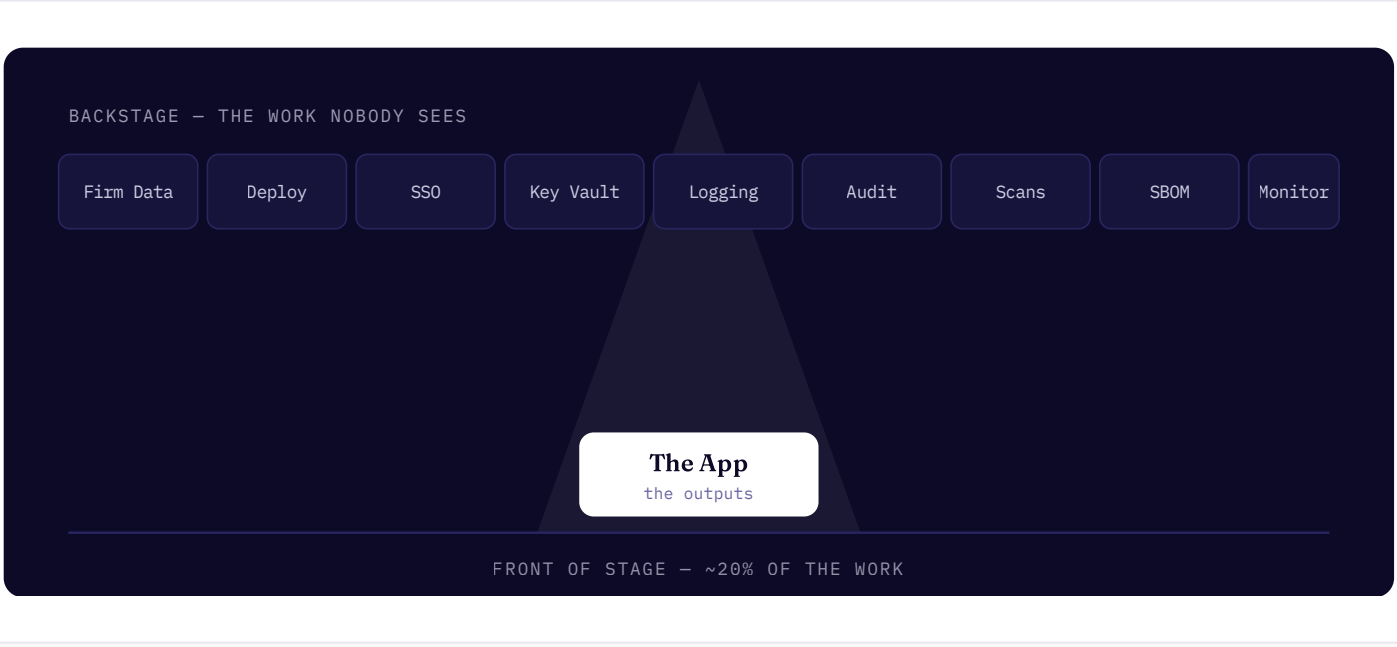

Figure 2. The app is the front of the stage. Data access, deployment, authentication, and governance are the backstage that makes the show possible.

The rest of this paper follows the two walls a builder runs into, in the order they hit them. First the **data wall**: swapping that manual upload for a live connection to the firm's real sources. Then the **deployment and governance wall**: getting the app shared, cleared by security, and signed off for compliance. Either one is usually enough to kill the project on its own.

> TAKEAWAY

> The walls are all backstage, and nobody hands the builder a map.

03 · WALL ONE, FROM A SPREADSHEET TO A LIVE SOURCE

Why is replacing the upload so much harder than building the app?

The spreadsheet upload is a dead end. Every refresh means exporting the file again by hand, so the numbers are out of date the moment they load. You cannot trust the app for anything that has to be current. The fix is to connect it to the firm's real, official source for that data. And that is where the first problem shows up, one that has nothing to do with code.

At most firms, nobody's job is to help a business user find the right source and the team that owns it. Ask a simple question, like which system holds the master copy of this client record, this position, this fee schedule, and often no one can tell you. And if the builder does track it down, the next step is a change and prioritisation process built for big quarterly projects, not for one person's afternoon app. The request goes into a queue and never comes back out.

> The hard part is almost never the connector. It is finding the source, finding who owns it, and getting through change control before the enthusiasm wears off.

> TAKEAWAY
>
> Wiring up the data is the small problem. Finding the right source and getting access to it is the wall.

04 · THE DATA WALL CHANGES SHAPE WITH FIRM SIZE

Why is data access hard for a large bank and a five-person RIA, for opposite reasons?

This is the part that "just connect your data" advice tends to skip. The barrier is real at every size of firm, but it is a different barrier each time, and a fix built for one size will not work for the others.

At a **large bank**, the data is spread across on-premises systems, cloud platforms, and SaaS tools. Each one is owned by a different IT team, and each has its own way of getting data in or out. The wall here is organisational. No single team can say yes, and the process meant to coordinate them all is slow by design.

At a **mid-market firm**, there are fewer silos, but there is still a formal process and a queue, and one business app rarely makes it to the front of that queue.

A **small advisory team** has almost no internal bureaucracy, which sounds like an advantage until you notice where the data lives. For a small RIA, the data that matters usually sits with outside vendors, and getting API access to it is its own fight. One firm we spoke to was quoted 35,000 dollars a year for API access to a financial planning tool. Its whole team of five advisors was spending about 25,000 dollars a year on that tool to begin with. Access cost more than the product did. For a small firm, that one number ends the project.

Firm size	Where the critical data lives	Primary barrier	Typical wait
Large enterprise / bank	On-prem, cloud, and SaaS across many teams	Organisational silos and change control	Months
Mid-market	Mixed internal systems, some SaaS	Formal delivery process and prioritisation queue	Weeks to months

Firm size	Where the critical data lives	Primary barrier	Typical wait
Small RIA / advisory	Third-party vendor platforms	API access cost and vendor dependence	Blocked on commercial terms

Table 1. Same wall, different shape. The data barrier a firm faces depends far more on its size and structure than on the app being built.

> TAKEAWAY
>
> There is not one data problem. A real fix has to work for the firm fighting internal silos and the firm fighting a vendor's invoice.

05 · WHY THE ANSWER IS A DATA LAKE, NOT A TANGLE OF PIPES

Should every source expose an API, an MCP server, and a feed of its own?

Say you can reach a source. There is still a second problem: the same source has to be reachable in a few different ways, depending on who is asking. A chat assistant or an agent needs an MCP server. A one-off lookup, like a client record from the CRM, needs an API. If the firm wants its own analytics environment, that needs a data feed. Asking every source system to support all of these formats is hard enough as engineering. As change management it is a nightmare, because each system belongs to a different team with its own release schedule and its own reasons to say "not this quarter."

So more and more firms stop trying to make every source serve every format. Instead they land the data once, into a **low-cost lake they own**, and work from there. The lake becomes the single place that hands out clean MCP endpoints, APIs, and feeds. You do not have to ask twelve different teams to each build three interfaces.

There is a cost trap on the other side of this too. Jumping straight to an expensive platform like Snowflake before you understand how the data gets used is premature. At that point you do not know what gets queried constantly and what gets touched once a quarter, so you cannot size the spend or justify it. Land the data cheaply first. Watch which patterns repeat. Then move only the heavy-use data into a high-performance environment, where the cost actually buys you something.

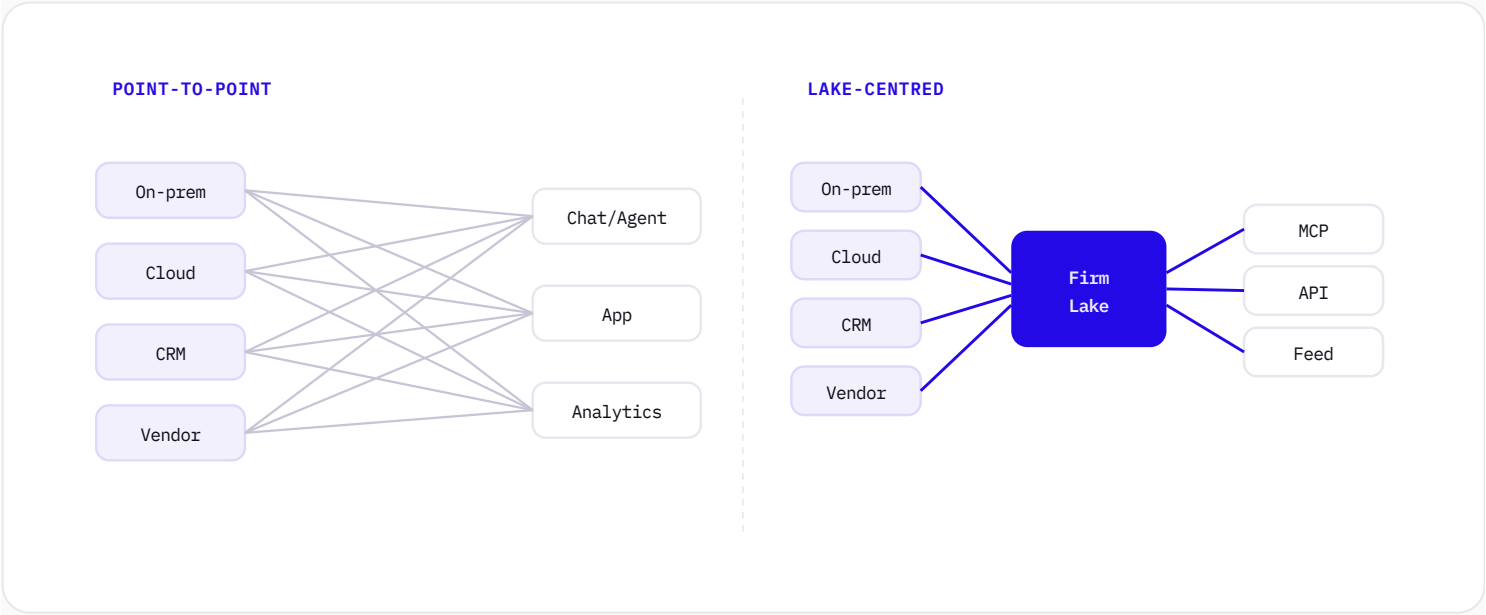

Figure 3. Point-to-point connectivity multiplies with every source and every consumer. A firm-owned lake collapses it into one place that publishes many faces.

> **TAKEAWAY**
>
> Land the data once in a cheap lake, serve every format from there, and pay for expensive performance only where the usage proves you need it.

06 · GOVERNANCE BELONGS AT THE LAKE, NOT INSIDE THE APP

Where do harmonisation, quality, access control, and PII masking actually live?

The lake is not just convenient plumbing. It is the one place where governance actually becomes manageable. Data has to line up across sources, so the same client, position, or fee means the same thing everywhere. It has to pass quality checks, so the app is not confidently wrong. It has to sit behind access controls, so people only see the rows they should. And it needs PII masking, so sensitive fields never reach an eye, or a model, that should not see them.

Try to enforce all four of those across a scatter of direct APIs and MCP endpoints, each wired into a different source, and you lose control fast. There is no single place to apply a rule, let alone prove you applied it. Putting harmonisation, quality, access, and masking in one place at the lake is a second reason to own it, on top of the connectivity reason from the last section.

This is where a platform-independent semantic data fabric earns its keep. **Clarista's fabric lets any source be brought in, governed, and published through MCP servers and APIs, and it stands up the firm's own data lake under the firm's control.** Harmonisation, quality, access, masking, and lineage all live in the fabric. Every app and agent downstream inherits them, instead of rebuilding them one connection at a time. For the longer version of this argument, running one governed data plane with zero data egress, see our companion paper on AI agent governance in financial services.

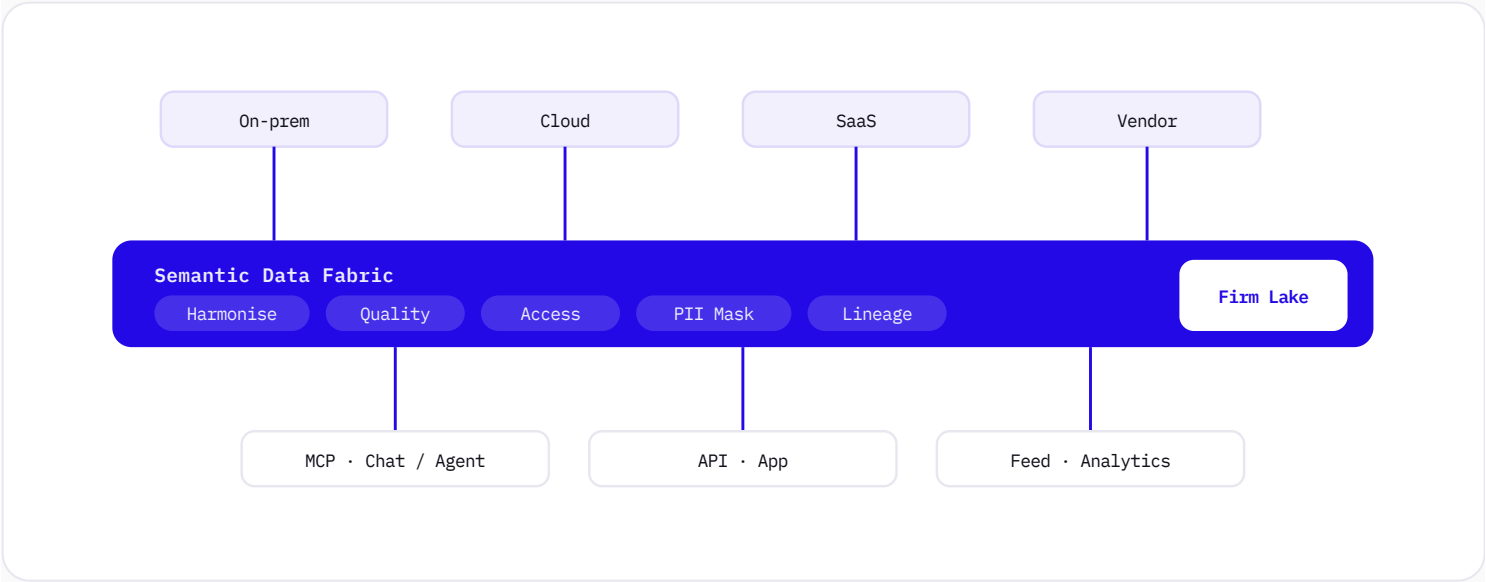

Figure 4. When every request passes through one governed layer, the app inherits harmonisation, quality, access control, masking, and lineage instead of reinventing them.

> TAKEAWAY
>
> Govern once at the lake and every app gets it for free. Govern connector by connector and you cannot really prove anything.

07 · WALL TWO, SHARING WITHOUT EMAILING A ZIP FILE

How do you get the app to your colleagues while still owning the edits?

Say the data problem is solved. The app now pulls live data from real sources, and the manual upload is gone. The builder is thrilled, and then hits the second wall right away: the app is still a file on one laptop. For colleagues to use it, it has to run somewhere everyone can reach. And the builder needs to keep editing that shared version without emailing a new copy every time.

Most business teams have no idea what "deploy it centrally" even means. So they go back to the same IT door they knocked on for data, and get the same wait. The only way IT knows to put software in front of a lot of people is its normal delivery and DevOps process, and that takes weeks or months, not a click. That process was built for planned systems with budgets and roadmaps, not for one analyst's app that needs to be live by Thursday.

> TAKEAWAY
>
> "Share it with the team" quietly means "deploy it like production software." That is a completely different job from building it.

08 · THE PRODUCTION CHECKLIST NOBODY HANDED THE BUSINESS

What does IT actually do to an app before it can go live?

Central deployment takes weeks because it is not one step. It is a checklist, and in a regulated firm most of it is non-negotiable. Just naming the list is half the point, because the business builder has never seen it.

Backstage requirement	What it is	Why it matters in a regulated firm
SSO	Single sign-on tied to corporate identity	Only the right people can open the app, and access is revoked the moment someone leaves
Key vault	Managed store for secrets and credentials	API keys and connection strings never live in the app's code or a config file
Code repo & version control	Tracked source with history and review	Every change leaves a diff, so the app can be reviewed, rolled back, and attributed
SAST	Static analysis of the source code	Catches insecure code before it ships, not after an incident
IaC & configuration	Infrastructure defined and versioned as code	The environment is reproducible and reviewable, not hand-built and undocumented
Image CVE scanning	Checks container images for known vulnerabilities	Prevents shipping an app on top of a component with a public exploit
DAST	Dynamic testing of the running web app	Finds vulnerabilities that only appear at runtime, on the live surface
SBOM	Software bill of materials	A complete inventory of components, so a future vulnerability can be traced in minutes

Table 2. The production-readiness checklist. This is the work IT performs, largely by hand, for every app before it is allowed in front of users.

This is the work IT does for weeks, on every app. Skip it and you do not have a shared application. You have another ungoverned end-user tool. In a regulated firm, an ungoverned tool that touches client data or drives decisions is not a convenience. It is a risk, and it comes with questions an examiner can ask. What decisions get made through this app? How is AI used inside it? Who can access it? And what happens when the person who built it leaves, taking the only real understanding of it with them? For more on the code and security side, see how Clarista handles code scanning and continuous compliance evidence.

> TAKEAWAY
>
> The gap between an end-user tool and a production app is a fixed checklist. Today a person works through every item by hand, for every app.

09 · THE SOFT WALLS, ROLE ANXIETY AND APP SPRAWL

If business teams start building apps, what happens to developers, and to governance?

Not every wall is technical, and the ones that are not kill just as many projects. Two of them come up at almost every firm.

The first comes from developers. If business teams can build their own apps, what is my job? It is a fair question. Brush it off and you get quiet resistance from exactly the people the builder needs to get past the deployment wall.

The second comes from leadership. If we let these little apps spread, we will have hundreds, then thousands, in no time. Who keeps track of them? Who governs them? That worry is fair when every app is a one-off, with its own data wiring and its own deployment, because then every app is a separate thing to secure, watch, and eventually retire.

Both worries have the same answer. Sprawl is only unmanageable when every app is unique. When every app pulls data from one governed fabric and ships through one standard pipeline, a thousand apps are not a thousand problems. They are one managed estate with a thousand entries in it. The developer's job changes too. Instead of hand-writing each app, they own the platform and the standards that every business-built app inherits. That is a bigger job, and a more durable one.

> TAKEAWAY
>
> The way to handle sprawl is not to say no. It is to make every app land in the same governed place by default.

10 · BREAKING THE WALLS, DATA PIPES PLUS ONE-CLICK DEPLOYMENT

What has to become automatic for a business builder to actually ship?

Put the two walls side by side and the answer is the same for both. Build the backstage work once, with the people who understand it, and then make it automatic for everyone who does not.

For the data wall, that means switching on governed data pipes through the fabric. Once they are on, the app stops relying on manual uploads. It shows the latest data from every source, already harmonised, quality-checked, access-controlled, and masked. The spreadsheet tab becomes a live connection, and the builder never exports a file again.

For the deployment wall, it means turning that checklist from Section 08 into **one click, for someone with no technical background, onto an environment IT set up once**. IT configures the sign-on, the key vault, the repository, the scanning rules, and the infrastructure standard a single time. After that, every app deploys against that standard on its own, running the whole hardening checklist without

the builder even knowing it is there. The business builds the front of the stage. The platform builds the backstage once, for everyone.

Figure 5. Two moves break both walls: governed data pipes replace the upload, and one-click deployment runs the full checklist against an environment IT configured once.

> **TAKEAWAY**
>
> The business builds the app. The platform builds the backstage once, for everyone, so shipping is a click instead of a project.

---

11 · CLARISTA

How Clarista gets AI-built apps to production in financial services

Everything in this paper lives in the gap between a working demo and a shared, governed, examinable system. Clarista closes that gap, on infrastructure the firm owns, with the data never leaving the firm's perimeter. Two parts of the platform map onto the two walls.

- **AI Data Fabric.** It reaches data where it already lives and handles sourcing, harmonisation, quality, access control, PII masking, and lineage, with zero data egress, on infrastructure the client owns. It stands up the firm's own low-cost data lake and publishes governed MCP servers and APIs from it, so every app and agent draws from one governed source instead of a tangle of point-to-point connections.
- **Deployment and Monitoring.** One-click deployment for non-technical builders, onto the environment IT set up once, running the full hardening checklist for them: sign-on, key vault, static and dynamic scanning, software bill of materials, infrastructure as code, and image vulnerability checks. Once the app is live, monitoring keeps watching, so a shared app stays a governed one.

It builds on itself. Data flows through one governed fabric, every app deploys hardened against one standard, and each new app adds to a managed estate instead of a pile of ungoverned prototypes. And it is running in production: Clarista supports financial-services firms live across several segments and continents, is SOC 2 Type II attested and ISO 27001 certified, and was named a WealthTech Americas 2026 winner.

No obligation. Bring an app you built with Claude Code or Codex that you have not been able to get past your data, deployment, or compliance walls.

FAQ

Frequently asked questions

Why can't I just share my AI-built app as a file with my team?

A file emailed or zipped around has no shared home, no shared login, and no way to update everyone at once. For colleagues to use it safely, the app has to run centrally, behind single sign-on, with logging and monitoring, so access is controlled and everyone is on the same version. That is the second wall builders hit, right after the data one.

How do I connect an AI-built app to my firm's real data instead of a spreadsheet?

You point it at the firm's real source instead of a manual export. The catch is finding that source and its owner, then getting a governed path to it. Most firms handle this by landing data into a low-cost lake they own and serving governed MCP and API endpoints from it, so apps read live data instead of a stale upload.

Why is API access to my financial planning or wealth tool so expensive?

A lot of vendors treat API access as a premium add-on, sometimes priced higher than the subscription itself. For a small advisory team, that can cost more than the tool does in the first place, which kills the project on price rather than on any technical problem. Owning your data in a lake you control cuts down on paying that toll again and again.

Do I need Snowflake to run AI apps on my firm's data?

Not at the start. Jumping to an expensive platform before you know how the data gets used is premature, because you cannot size or justify the spend yet. Land the data cheaply first, see what gets used heavily, then move only that data to a high-performance environment where the cost pays off.

What does it take to deploy an AI-built app in a regulated firm?

Deploying to production is a checklist, not a single step: single sign-on, a key vault for secrets, version control, static and dynamic code scanning, infrastructure as code, image vulnerability scanning, and a software bill of materials. IT usually works through all of it by hand for each app, which is why it takes weeks. Automate that checklist against an environment IT sets up once, and it turns into a click.

Who is responsible for governing all the AI apps my business teams build?

Sprawl is only a problem when every app is a one-off. When every app pulls data from one governed fabric and ships through one standard pipeline, a lot of apps become a single managed estate instead of a pile of separate risks, and the platform team owns the standard they all inherit.

Is vibe coding safe for regulated financial firms?

Vibe coding is safe when the output ships through controls, not around them. The risks are unscanned code, secrets in files, and firm data pasted into unmanaged tools. A governed pipeline fixes this: every AI-built app gets code security scans, dependency checks, single sign-on, logging, and governed data access before a second user ever touches it. Vibe coding security is a pipeline property, not a builder skill.

How do firms stop AI-built apps from becoming shadow IT?

Not by banning citizen developers. Shadow IT grows where the sanctioned path is slower than the workaround. Give builders a paved road: one governed data fabric, one one-click deployment pipeline, automatic logging and audit. Then

hundreds of apps become a single managed estate, and AI data governance is enforced by the platform instead of policy memos.

SOURCES

References and further reading

- Clarista, What is vibe coding. clarista.io/blog/what-is-vibe-coding
- Clarista, The best vibe coding tools of 2026. clarista.io/blog/best-vibe-coding-tools-2026
- Clarista, AI Agent Governance: Agents vs AI-Built Apps in Financial Services. clarista.io/whitepapers/ai-agent-governance-financial-services
- Clarista, AI Data Fabric: governed data access for AI through one MCP door. clarista.io/platform/data-fabric
- Clarista, code scanning and continuous compliance evidence for AI-built apps. clarista.io/platform/code-security
- Adoption figure: more than four million people now build software with AI assistance, reported across industry press. Treat as directional.

S

Suvrat is the founder and CEO of Clarista. He spent more than 25 years in enterprise data leadership, including as founding Chief Data Officer at UBS Asset Management, with earlier roles at Credit Suisse and Morgan Stanley. He holds an MBA from Columbia Business School and writes on governed AI deployment for regulated financial-services firms.