Highnote

Built To Launch: Finally, A Payments Platform Built For You.

Highnote builds around your business so you can launch faster, differentiate the experience, and keep innovating as you scale. This is the engineering playbook for the teams making it happen.

Table of Contents

Legacy infrastructure holds engineering teams back at every stage: from the first sandbox request to the first live transaction and every product iteration after. This playbook is for the engineering leaders who are done accepting that as inevitable. It makes the case for a platform built around your roadmap, your customers, and your vision, not someone else's predefined constraints.

Foreword

The embedded finance market has never moved faster. Fintechs are emerging overnight. Enterprises are launching financial products as core differentiators. Customer expectations are set by the most capable consumer apps ever built. Yet the infrastructure most engineering teams are working with was designed for a completely different era.

The result is predictable. Launches that drag on for months. Migrations that become year-long projects. Product iterations that require rearchitecting entire systems. Engineers whose most valuable time goes toward fighting a platform rather than building for their customers.

This playbook is for engineering leaders who have seen that movie before and are ready for a different one. CTOs, Heads of Engineering, and technical product leaders who understand that the right infrastructure is a strategic advantage, and the wrong one is a ceiling.

We will cover why launches take as long as they do, what changes when you build on a foundation designed for speed and flexibility, and what it looks like in practice when the infrastructure gets out of the way and lets great engineers do their best work.

We hope you partner with Highnote. But whether you do or not, this playbook will change the way you evaluate your infrastructure choices.

Happy reading.

Built for You. Three ways.

Everything Highnote builds comes back to the same idea: most platforms make you fit their box. Highnote is built to fit yours. That shows up across three capabilities that compound into a durable competitive advantage.

Built to Launch Faster	Built to Differentiate	Built to Innovate
Less friction, fewer blockers, and a launch that holds up after go-live.	Branded experiences and flexible program design that competitors cannot match.	Add new products and capabilities without a rewrite. Your roadmap, your pace.

The True Cost of Legacy Infrastructure

When a company decides to launch a payment or card program, the question should be: what will this program do for our customers? In practice, the question tends to become: why is this taking so long? The industry has normalized timelines that have nothing to do with the complexity of the work and everything to do with the limitations of the platforms underneath it.

Legacy platforms force engineering teams to absorb costs that should never exist. The problems are structural, they compound over time, and they are entirely avoidable with the right infrastructure.

Platforms built for a different era

Many of the processing platforms in production today were architected ten to twenty years ago. They pre-date modern API design, cloud-native architecture, and any serious consideration of developer experience. The result is platforms that operate through rigid, predefined configurations.

> A lot of these platforms have been built a decade or two ago, so they are not flexible in any way. They have these predefined buckets. If you don’t fit in the bucket, you are going to have to change your program or go find someone who can support you.”
>
> Joe Feltis, Head of Implementation Engineering, Highnote

For engineering teams, this rigidity has a direct cost. Every use case that falls outside the platform's assumptions requires workarounds or custom integrations. Every new capability requires an additional vendor relationship. Every additional vendor adds integration work, compliance exposure, and ongoing maintenance. The burden doesn't stay the same over time. It grows.

Broken before the build starts

Even the evaluation phase is slow. Getting a sandbox environment from a traditional provider means scheduling a call, waiting for manual provisioning, and then working within a configuration that someone else defined. Documentation is often a reflection of what the platform could do years ago, updated infrequently if at all.

> Here's documentation that was built on a monolith legacy platform, designed in 1965, updated in 1987, and not touched since. Access and provisioning of sandbox environments needed to be created by me. I had to go in and create a demo environment, stand it up, put all the parameters I thought were relevant. They could not change them. If they needed a change, it had to come to me."
>
> Tom Chiesa, Solutions Engineering, Highnote

At Highnote, a developer can go to highnote.com, sign up, provision a sandbox, create a card product, and begin testing without a single call and without waiting for anyone. The first touchpoint with the platform compresses the entire evaluation and pre-integration phase.

Documentation is often treated as a secondary concern, but for developers it is the primary interface to a platform. Stack Overflow's 2024 Developer Survey found that 90% of developers rely on API and SDK documentation as their primary resource when learning and integrating with new systems. When that documentation is incomplete or outdated, the entire development process slows.

The cost of poor documentation is measurable. Postman's 2024 State of the API Report found that 39% of developers cite inconsistent or outdated documentation as the biggest obstacle to effective API collaboration. What appears to be a minor inconvenience becomes a structural bottleneck to speed.

The compounding cost of fragmentation

Fragmented infrastructure multiplies every problem. When issuing, ledgering, compliance, acquiring, and reporting live across separate vendors, every transaction touches multiple systems. Every new capability requires a new contract and a new integration. Reconciliation becomes a manual exercise. Audit preparation becomes an extended project. Engineering overhead compounds at every stage, from launch through every subsequent iteration.

The platforms doing this to teams were built to serve the bank's workflow, not yours. Highnote was built differently.

Engineering time absorbed by the platform

The friction caused by legacy infrastructure is not theoretical. Atlassian's 2024 developer experience research found that 69% of developers lose eight or more hours per week to inefficiencies such as technical debt, poor documentation, and fragmented systems. That is effectively an entire workday lost every week, not to building products, but to navigating infrastructure that was not designed with developers in mind.

The impact goes beyond productivity. The same research found that 63% of developers say developer experience plays a significant role in whether they stay in their current role. Poor infrastructure does not just slow teams down. It actively drives talent away.

9–12 Months

The time it typically takes to launch a card program on a legacy platform. That timeline is not driven by the complexity of the work. It is driven by the limitations of the infrastructure.

The competitive frame worth knowing

Legacy and rigid platforms were built for banks. Their constraints, their workflows, their compliance posture. Highnote is built for the companies building on top of financial infrastructure. That distinction shapes every architectural decision we have made.

Key Takeaways

- Legacy platforms built a decade or more ago carry structural constraints that slow every stage of a program, from evaluation through every post-launch iteration.

- Manual sandbox provisioning and outdated documentation create friction before a single line of production code is written.

- Vendor fragmentation multiplies integration complexity and turns ongoing maintenance into a permanent drag on engineering capacity.

- The platforms creating this problem were built for a different customer. Your team is not that customer.

Built to Launch Faster

Speed matters. The companies that launch faster enter markets earlier, capture feedback sooner, and build operational confidence before their competitors have shipped. Speed compounds: teams that move faster at launch continue to move faster afterward.

Speed is one dimension of what Highnote delivers. It sits alongside completeness, differentiation, and the ability to innovate continuously. Engineering leaders do not have to choose between a platform that moves fast and a platform that moves right. The architecture that enables completeness is the same architecture that enables speed.

Completeness is what makes speed real

> Speed is important. I don’t think it’s the most important. It’s probably a close second to completeness. If we could only do ten percent of what you want to offer but we can move really quickly, that would be irrelevant. For most of our subscribers, speed is very important, but the completeness of the offering is also very important. We want to move fast and launch quality products.”
>
> Joe Feltis, Head of Implementation Engineering, Highnote

Highnote was designed with this in mind. A platform built to launch quickly on an incomplete foundation creates a different kind of technical debt: one where every subsequent capability requires reworking what was already built. Highnote’s completeness means the foundation supports everything that comes after it without rework.

Technical leaders are evaluating earlier

The people making infrastructure decisions have changed. CTOs and Heads of Engineering are now at the table earlier, and they come with specific technical questions. What are the capabilities? How complex is the integration? What does the developer experience look like? These questions used to come after a commercial agreement. They now come first.

> The technical piece is shifting very much to the front end of the conversation. Decision makers on the subscriber side are technical people. CTOs, heads of engineering. They want to see what the capabilities are, how to solve use cases with your platform, how difficult it is to implement. All of those types of questions come out very early.”
>
> Joe Feltis, Head of Implementation Engineering, Highnote

This shift raises the stakes for platform evaluation. Engineering leaders need to know, before a contract is signed, whether the platform they are choosing will let them move at the pace their business requires. Platforms that cannot answer that clearly, from the first technical conversation, lose the evaluation before the proposal is written.

The platform expands what you thought was possible

A consistent pattern in early Highnote engagements: subscribers arrive with one vision for their program and leave with a larger one. The platform’s breadth across commercial prepaid, consumer cards, fleet, credit, acquiring, payouts, and digital wallets creates possibilities that a vendor catalog does not reveal.

> A lot of times they will change what they want their offering to be midway through just by exploring our platform. They were expecting to need to do something in-house, and then we can say: we have that. It cuts significant time off their implementation because they were expecting to build it themselves.”
>
> Joe Feltis, Head of Implementation Engineering, Highnote

The discipline is to stay focused on the initial use case, launch cleanly, and expand from a strong foundation. That sequence, launch with confidence and then innovate continuously, is exactly what the platform is designed to support.

The stakes are rising quickly. Embedded finance is no longer an emerging category. It is becoming core infrastructure. Grand View Research estimates the market was valued at $83.3 billion in 2023 and is projected to reach $588.5 billion by 2030, growing at a 32.8% compound annual growth rate. As the opportunity expands, the cost of slow infrastructure compounds alongside it.

$588.5 Billion

Projected global embedded finance market size by 2030, up from $83.3 billion in 2023 at a 32.8% compound annual growth rate. The infrastructure decisions made today will determine who captures this opportunity.

Source: Grand View Research, 2024.

In traditional environments, launching a financial product can take nine to twelve months when accounting for integration complexity, vendor coordination, and compliance overhead. Teams operating on modern, unified platforms are able to compress those timelines dramatically, often moving from sandbox to production in a matter of weeks, depending on the scope of the program. This is not a function of working faster. It is a function of removing the structural barriers that slow teams down in the first place.

Key Takeaways

- Speed matters. The companies that launch faster stay faster. Post-launch velocity compounds with every iteration.

- Speed and completeness come from the same source: an architecture designed from the ground up to support the full range of what a program needs - and allow it to expand without requiring a rebuild.

- Engineering leaders are evaluating platforms earlier. Developer experience is a front-line competitive differentiator, not a post-sale detail.

- Most programs expand in scope once subscribers understand what the platform can do. That expansion is a feature, not a problem, when the architecture supports it.

The Proof:
Real Stories from the Field

The most convincing evidence for any infrastructure claim is not a benchmark. It is a program that shipped, served real customers, and did so on a timeline that the team had not thought was achievable. Below are three stories that demonstrate what is possible when the right platform removes the right friction.

ferry

Ferry: Instant Payments, Dramatically Faster

Ferry had a working setup. They used Highnote for fund aggregation alongside a third-party provider to push funds externally to cardholder accounts. The workflow functioned, but it added vendor complexity, additional integration surface, and ongoing cost.

When Highnote launched Instant Payments, enabling Highnote financial accounts to push funds directly to external payment methods using Visa and Mastercard rails, Ferry had an opportunity to simplify their architecture significantly. The implementation path was deliberately clean: three mutations replaced a multi-step cross-vendor workflow. Token generation. Payment method tokenization. Transfer initiation.

> “It’s a very complex solution solved by a very simple implementation. We internalize the complexity so they don’t have to. It’s literally a couple of mutation calls to facilitate this money movement. That is what allowed them to move so quickly.”
>
> Joe Feltis, Head of Implementation Engineering, Highnote

Ferry's original rollout plan was staged for gradual adoption. After the initial testing phase, they had seen enough. They released the guardrails entirely. The migration from their existing setup to a fully Highnote-native Instant Payments flow concluded faster than any comparable migration would have on a legacy platform, and faster than Ferry had anticipated.

Highnote in action: Ferry

- Program: Express Pay tip disbursements and cardholder fund transfers.
- Challenge: Multi-vendor workflow adding integration complexity and operational cost.
- Solution: Instant Payments, enabling direct fund movement from Highnote accounts to external cards via Visa and Mastercard rails.
- Key enabler: Three mutations replaced what had been a multi-leg, cross-vendor process. The platform internalized the complexity.
- Outcome: Simplified operations, reduced vendor dependency, and lower cost, with no disruption to active cardholders.

MUDFLAP

Mudflap: Enterprise Fleet, Built to Grow

Fleet payments are technically demanding. Level 2 and Level 3 data from fuel pump terminals, authorization flows with advice messages, high-volume commercial transaction reconciliation. There is no lightweight version of this use case.

Mudflap needed a partner that could build alongside them and keep pace as their requirements matured. Highnote built the fleet solution in close partnership with Mudflap, assembled the initial program quickly enough to get them to market, and has iterated alongside them as the program has grown in capability and complexity.

> We built our fleet solution for Mudflap and we were able to put everything together really quickly for them to get out the door. And then we have iterated over time to make sure they are getting all of the L2/L3 data, the authorization with advice messages from the terminals on the pumps. We have been in really close partnership throughout the entire engagement."
>
> Joe Feltis, Head of Implementation Engineering, Highnote

The Mudflap engagement is an example of what Highnote means by continuous partnership. The program launched on a foundation capable of supporting what came next. Capability was added over time without rebuilding what already worked. The architecture held.

Highnote in action: Mudflap

- Program: Enterprise fleet payments platform for independent truckers and small fleet operators.
- Challenge: A technically demanding use case requiring Level 2 and Level 3 fuel pump data, authorization flows with advice messages, and high-volume commercial transaction reconciliation.
- Solution: A purpose-built fleet program developed in close partnership with Mudflap, assembled quickly for initial market entry and iterated continuously as program requirements matured.
- Key enabler: Backwards-compatible architecture allowed new capabilities to layer in without rebuilding the existing foundation. Every iteration added to what was already working.
- Outcome: Active and growing market presence, with ongoing capability expansion supported by a continuous Highnote engineering partnership.

Fillip: Full Stack Without a Separate Processor

Fillip had built payment infrastructure for the Canadian market. Entering the US required a different set of capabilities, and the question was whether to build and certify a standalone processor or find a partner that already had what they needed.

The requirement list was comprehensive: physical cards, virtual cards, digital wallet tokenization, commercial prepaid, fleet, Level 2 and Level 3 transaction data, and full reconciliation and reporting. A list that comprehensive typically generates months of scoping work before integration begins.

> Instead of building and certifying their own processor for the US market, they said: we are going to do it through you. Because you have the ledger, the capability for physical, virtual, digital wallet tokenization, commercial prepaid, fleet, Level 2 and Level 3 data. We don’t need to stand up a separate processor. That was really gratifying to say: yes, we can do all of that. You don’t need to do it all on your own.”
>
> Tom Chiesa, Solutions Engineering, Highnote

Fillip’s US market entry required a single integration. Not a processor build. Not a multi-vendor coordination project. One platform, covering the full requirement stack.

Highnote in action: Fillip

- Program: US market entry for an established Canadian fleet payments company.
- Challenge: Full commercial payments capability required across multiple card types, Level 2/3 data, and reconciliation, without building a standalone processor.
- Solution: A single Highnote integration covering the complete requirement set.
- Key enabler: Unified platform with built-in ledger, fleet support, and physical/virtual/digital wallet capabilities under one program.
- Outcome: Faster US market entry, lower infrastructure complexity, and a single source of financial truth from day one.

What these stories have in common

Three different programs, three different use cases, three different starting points. What they share is architecture that internalized complexity so the subscriber did not have to. The platform removed friction that would have cost months on older infrastructure. The teams built what they came to build, and launched.

Key Takeaways

- Ferry migrated to Instant Payments faster than any comparable migration would have allowed on a platform that hadn’t been built with flexibility as a core design principle.

- Mudflap launched a technically complex fleet program quickly and has expanded it continuously without rebuilding the underlying architecture.

- Fillip entered the US market with a single integration covering their complete capability stack, eliminating the need to build and certify a standalone processor.

- The consistent thread: when the platform internalizes complexity, subscribers ship faster and iterate from a stronger foundation.

Built to Differentiate: The Infrastructure Behind the Experience

Faster launches and better subscriber experiences do not happen because a team works harder. They happen because the infrastructure allows it. Every architectural decision Highnote made from day one was a decision about what kind of experiences should be possible, and what should never require a workaround.

Self-service from the first interaction

Everything you need to evaluate, prototype, and build before the first conversation. A developer evaluating Highnote can go to highnote.com, create an account, provision a sandbox environment, create a card product, and begin testing without scheduling a call, waiting for manual provisioning, or working within parameters someone else defined.

This matters for reasons that go beyond convenience. It compresses the evaluation phase and means that engineering teams arrive at the first implementation conversation with real, hands-on platform context. The first interaction with Highnote is with the platform itself. That creates a completely different kind of confidence.

GraphQL: built for how developers actually work

Highnote's API is built on GraphQL. This is a deliberate architectural choice with direct implications for how quickly teams can build, how much data they need to manage, and how easily they can iterate.

On a REST-based platform, creating a card program requires separate calls to separate endpoints. Each endpoint has its own parameters, its own response shape, its own failure modes. Developers must orchestrate across multiple calls to accomplish a single outcome. GraphQL consolidates this. One endpoint. Mutations that compose naturally. Request exactly the data you need and receive exactly that.

> In Highnote, it’s mutations that sit on the same endpoint. It’s just a simple POST. You don’t have to make a ton of changes. I’m using the same key. It is very easy to articulate. You can script it through any AI coding tool and in a matter of minutes, because it is so simple to understand the way it is architected, it can do it quickly.”
>
> Tom Chiesa, Solutions Engineering, Highnote

For teams already experienced with GraphQL, the learning curve is minimal. For those coming from REST-based platforms, it is short.

> We get engineers comfortable with GraphQL in a single session. We walk them through what the value is and what the differences are, and it just clicks.”
>
> Joe Feltis, Head of Implementation Engineering, Highnote

Developer expectations have shifted

The broader shift toward API-first development has raised the baseline for what engineering teams expect from any platform they evaluate. Postman’s 2024 State of the API Report found that 74% of organizations now describe themselves as API-first, up from 66% the prior year. And 63% of developers report they can produce an API in under a week. Speed is no longer a differentiator in this context. It is the baseline expectation. Platforms that cannot meet it create friction at the most critical stages of evaluation and implementation.

Documentation that lives with the platform

For legacy platforms, documentation is a tax. It reflects what the platform could do at the time someone last updated it, which is rarely now. Highnote treats documentation as a product, maintained continuously and updated as the platform evolves.

> We have folks continually reviewing our documentation. We have a channel where you can say: I identified something that might be different now, and it gets updated right away. The way that we stay on top of things is to ensure that what we put out into the world is what people can actually do.”
>
> Tom Chiesa, Solutions Engineering, Highnote

When an error code appears, the documentation explains what it means and how to resolve it. When a new capability ships, the documentation is ready. Engineers resolve problems independently, without submitting a support ticket and waiting. Self-sufficiency is the design intent.

This is where most legacy platforms fall short. When documentation lags behind the product, or when developers must rely on support teams to move forward, speed breaks down. Modern platforms invert that model. Documentation becomes a living part of the product, continuously updated and directly aligned to what developers can actually build.

This design philosophy is validated by the broader research on high-performing engineering teams. Google Cloud’s 2024 DORA research highlights that high-performing engineering organizations prioritize user-centered platform design, building systems that developers can use independently rather than relying on internal support layers. The more self-sufficient developers are, the faster their teams move.

A ledger built first, not retrofitted

Among the most consequential architectural decisions at Highnote was building the ledger first. A purpose-built, double-entry ledger is the foundation of every program on the platform. Every transaction, every fund movement, every financial event is tracked against a single source of truth.

At older platforms, the ledger was an afterthought. It was added after the transaction processing core was built, which meant it was always trying to catch up. The consequences show up in reconciliation complexity, in reporting delays, in financial data that requires cleaning before it can be trusted.

> Ledger was not the first thing at platforms built before us. It wasn't even the second or third thing. It was: oh yeah, we have to track the money too. And they ran into problems being able to reflect things outside of transactional activity in an accurate fashion."
>
> Tom Chiesa, Solutions Engineering, Highnote

For engineering and finance teams alike, a purpose-built ledger means real-time financial visibility without extraction scripts, without batch processes, and without reconciliation work that never quite finishes.

Insert, not rewrite

The architectural phrase that captures Highnote's approach to feature expansion is this: it is an insert, not a rewrite.

> Our core objective is to add new without creating a need for you to recode or redevelop. It's a new feature set, incorporate it. Don't rebuild. If you came in wanting commercial prepaid and decide you want a charge product, there are some tweaks to make, but it is not a show-stopper. It is not go back to zero."
>
> Tom Chiesa, Solutions Engineering, Highnote

On architecturally sound, backwards-compatible infrastructure, adding new capabilities means inserting new mutations. The existing workflow continues. The new capability layers in. Teams on legacy platforms are constantly paying down technical debt with every release. Teams on Highnote compound their investment.

Key Takeaways

- Self-serve sandbox access means the first interaction with Highnote is with the platform itself. Engineering teams arrive at implementation conversations with real context.

- GraphQL architecture reduces endpoint complexity and lets developers request exactly the data they need from a single interface.

- Continuously maintained documentation means engineers can resolve problems independently, in context, without waiting for support.

- A purpose-built, double-entry ledger provides real-time financial accuracy from the first transaction, not from a batch process the next day.

- Backwards-compatible feature expansion means new capabilities are added to existing programs. The word is insert. Not rewrite.

Built to Innovate: How Programs Evolve After Launch

Launch is the starting line. The competitive advantage that compounds most meaningfully is what happens after it. Teams that built on the right foundation iterate continuously. Teams that made different choices spend their post-launch energy managing the weight of those choices.

Post-launch velocity compounds

Every feature that ships without requiring a rebuild creates capacity for the next one. Every new capability that does not require a new vendor creates more room for product differentiation. Over time, the gap between teams that built on complete, flexible infrastructure and those that did not widens in ways that are very difficult to close.

Highnote's technical partnership model is built to support this compounding. Implementation engineering teams, all career engineers, stay engaged with subscribers as their programs evolve. The relationship does not end at go-live. It deepens.

> The depth we are able to have on the post-sales side, covering security, resiliency, and things adjacent to the implementation that they may not be thinking about yet, that is where the experience really stands apart from anything else in the market."
>
> Joe Feltis, Head of Implementation Engineering, Highnote

Real-time visibility for real-time decisions

Every transaction event, ledger update, fee calculation, and dispute workflow on the Highnote platform is accessible via webhook the moment it happens. Clearing and settlement run multiple times daily. Visa and Mastercard each settle up to four times per day, so data reflects reality in near real time rather than waiting for a nightly batch process.

> If it happens within the platform, you can receive a webhook for it. You don't have to wait until the end of the day or the next processing day. As we get it, it is going to be pushed out to you. It is there when you need it."
>
> Tom Chiesa, Solutions Engineering, Highnote

For product teams, this means building features on accurate, current data. For finance teams, it means the numbers reflect reality, not a reconciliation cycle completed hours or days ago. For operations, it means visibility into what is happening now, not what happened yesterday.

Expand programs, your roadmap, your pace

Subscribers consistently find that their vision for what their program can become grows once they start working within the platform. Commercial prepaid, consumer cards, fleet, credit, acquiring, payouts, digital wallets: the full capability stack is available through a single integration, on the same ledger, under the same program.

Adding a new product line, expanding to a new market, or launching a new card type does not require rearchitecting what was already built. It requires adding to it. The architecture was designed from day one to support this.

> Issuing, built for you. Acquiring, built for you. Payments, built for you. Platform, built for you. You can cycle in whatever word is descriptive of what the product needs to do. And it is built for you."
>
> Tom Chiesa, Solutions Engineering, Highnote

Your roadmap, not rented.

On legacy platforms, the product roadmap is constrained by the platform roadmap. Features ship when the vendor ships them, on the vendor's timeline, within the vendor's architecture. Highnote is built to move with your roadmap. New capabilities integrate forward. They do not require starting over.

Key Takeaways

- Post-launch velocity compounds. Teams building on complete infrastructure iterate faster with every release, not slower.

- Real-time webhooks and data access mean product, finance, and operations teams are working with current information, not last night’s batch extract.

- Platform breadth means programs expand into new products and markets without rearchitecting. The word is add, not rebuild.

- Highnote’s engineering partnership deepens after launch. The technical relationship is designed for the long arc of a program, not just the first go-live.

Built for You: The Team Behind the Platform

The right infrastructure is foundational. The people who help you build on it are equally important. Highnote made deliberate choices about how to build its technical teams, and those choices show up in every subscriber engagement, from the first technical conversation through every post-launch iteration.

Engineers with people skills

The standard model for technical sales and implementation teams is to hire people with strong commercial skills and give them enough technical training to navigate basic questions. Highnote built the inverse of that.

> We have flipped the traditional approach. We are hiring engineers who happen to have people skills. From the sales engineering function to the implementation engineering function, we are all career engineers. We have built full-stack applications. We have worked in professional services and consulting. We have built these exact implementations that our customers are building. We can provide really technical, detailed responses. We can give them sample code, sample operations for the GraphQL API."
>
> Joe Feltis, Head of Implementation Engineering, Highnote

The effect is visible from the first technical conversation. Engineers on the subscriber side, and increasingly the decision-makers are engineers, get substantive depth rather than escalations. Questions are answered in the room. Sample code exists. The engineering surface of Highnote's team meets the subscriber where they are.

White-glove to self-sufficient

The goal of Highnote's implementation engagement is not to create ongoing dependency. It is to transfer capability. The team moves fast alongside subscribers during implementation, and by launch, subscribers are equipped to operate independently.

> We want them to be as self-service as possible. We want you to be self-sufficient, because at some point you are going to launch and go into production. We want to limit that need. Make you, for lack of a better term, dangerous enough to manage and run this on your own.”
>
> Tom Chiesa, Solutions Engineering, Highnote

Quick-start templates, exhaustive error code documentation, and contextual guidance throughout the integration ensure that by the time a program goes live, the subscriber team knows the platform. Support is available when needed. But the design intent is that it will be needed less and less over time.

Parallel tracks, no bottlenecks

At some providers, one contact manages the entire implementation: solution definition, sandbox setup, compliance documentation, testing coordination, and program onboarding. One person responsible for everything eventually means one person who is overloaded, and that overload appears directly in the timeline.

> At older platforms I have worked at, I was solution engineer, delivery manager, and I supported program onboarding, all at the same time. I worked 14 and 15 hours a day. Great, you have one person you can count on. That one person is going to get overloaded and burned out. Here, we have people focused on specific tracks pushing things forward concurrently. You do not have to wait for one track to finish before the next one starts.”
>
> Tom Chiesa, Solutions Engineering, Highnote

At Highnote, implementation work moves on parallel tracks. Compliance, network operations, and technical implementation run concurrently. There is no single-threaded bottleneck. Progress on one track does not wait for progress on another.

Above all else: protect the subscriber experience

> Our tagline internally for technical sales has always been: above all else, protect the subscriber experience. Every interaction, internal or external, that is our main focus. If that is not exactly built for you, I don't know what is."
>
> Joe Feltis, Head of Implementation Engineering, Highnote

That principle shapes how Highnote measures the success of every implementation. The benchmark is not whether deliverables were checked off on schedule. It is whether the subscriber team leaves fully equipped to run their program, scale it, and grow it on their own terms. The entire engagement is designed around that outcome.

What to watch for when evaluating partners

If you are evaluating embedded finance providers, there are phrases that should prompt a much closer look. These are drawn from the experience of people who spent years at legacy providers and older platforms before coming to Highnote.

If they say this...	Here is what it means
"We will need to slot you in for integration."	Your implementation is a queue item. Expect to wait for someone's calendar before your program can move.
"You will need to re-architect for that change."	The platform was not built for flexibility. Every pivot will be expensive, every time.
"Each product requires a separate program and login."	Your operations team will find this unmanageable very quickly.
"You will need to maintain your own ledger."	There is no reliable source of truth on their platform. Reconciliation will be a permanent exercise.
"Let me connect you with our compliance team, then our bank team, then our implementation team..."	You are being routed across separate organizational silos. No single team owns your implementation outcome. Coordination across those groups becomes your responsibility for the duration of the program.

Key Takeaways

- Highnote's implementation teams are career engineers, built for technical depth from the first conversation through every post-launch iteration.

- The engagement is designed to build subscriber capability, not dependency. By launch, your team knows the platform.

- Parallel-track implementation eliminates the single-contact bottleneck that slows programs on platforms where one person owns everything.

- The warning signs are consistent. If a vendor is already creating friction before the contract is signed, that friction will not improve after it.

The gap between platforms is no longer measured in features alone. It is measured in how quickly teams can move from idea to execution. As developer expectations shift and embedded finance becomes core infrastructure, the ability to launch quickly and iterate continuously is becoming one of the most important factors in platform selection.

Built for You.

The premise of this playbook is straightforward: the companies that launch faster win. But winning is about more than speed.

It is about building on infrastructure designed around your business, your roadmap, and your customers. Infrastructure that launches faster because it was built to be complete. That differentiates because it was built to be flexible. That keeps innovating because it was built for evolution, not replacement.

Ferry migrated Instant Payments faster than any comparable migration would have allowed. Mudflap built a complex fleet program and has expanded it continuously. Fillip entered the US market with a single integration covering their full capability stack.

These are not outliers. They are what happens when the right engineering team builds on the right foundation.

Built to Launch Faster.

Built to Differentiate.

Built to Innovate.

Built for You.

When you are ready to see what your roadmap looks like on infrastructure built for it, we are ready to show you.

Contact Us →