PRODUCT MANAGEMENT CASE STUDY

Reimagining Enterprise Aadhaar Verification

How I transformed a legacy onboarding product into a configurable enterprise solution — improving adoption, developer experience, and verification success.

EXECUTIVE SUMMARY

The Challenge and the Opportunity

The existing DigiLocker product had reached a point where it was becoming difficult to scale. Its rigid workflows, limited configuration options, and legacy architecture made it harder for enterprise customers to integrate and customize the product to their needs. These limitations also increased the effort required from engineering teams to support customer requests. The goal was to modernize the platform into a more flexible, reliable, and configurable solution that could deliver a smoother experience for both customers and internal teams.

+7%

Verification Success Rate

Improvement achieved post-redesign

0

Downtime During Migration

Zero-downtime rollout achieved

7

Key Product Decisions

Strategic product choices made

4

Improvement Categories

Platform, DX, Reliability, Flexibility

DISCOVERY

Understanding Before Building

Before thinking about solutions, I wanted to understand where customers were getting stuck. Conversations with enterprise customers, support teams, and production data consistently pointed to the same issues - onboarding flows were too rigid, even simple configuration changes needed engineering support, integrations lacked flexibility, and it was difficult to understand why verifications failed. These insights became the foundation for the product improvements.

Customer Feedback

Structured interactions with enterprise implementation teams to surface recurring friction points.

Production Gap Analysis

Analyzed failure rates, support tickets, and drop-offs to uncover recurring customer pain points.

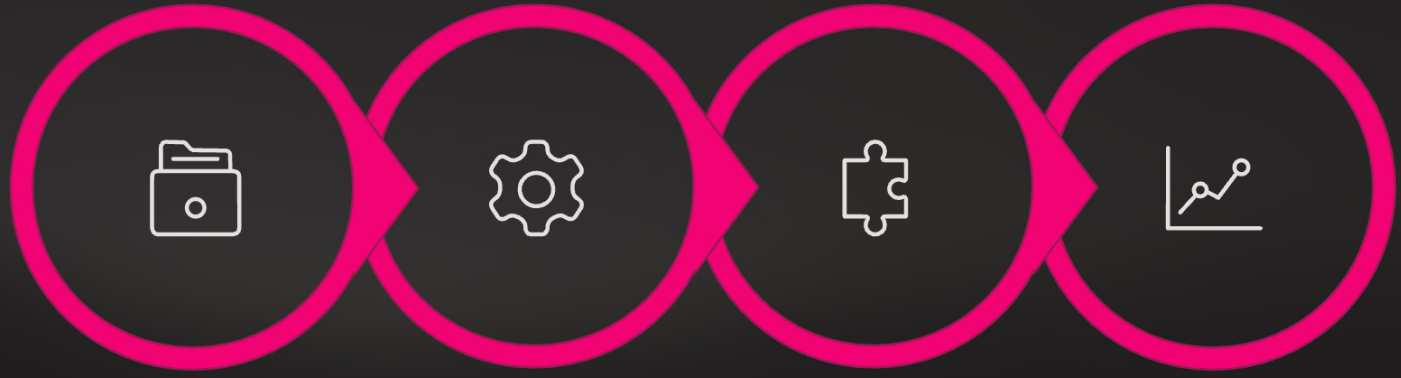
API Usability Analysis

Audit of integration patterns to identify where developer effort was highest.

PRODUCT VISION

From a Fixed Workflow to a Configurable Solution

The goal wasn't to add another feature - it was to redesign the product so it could adapt to different enterprise needs instead of forcing every customer through the same workflow.



This transformation required rethinking architecture, configuration, developer experience, and migration strategy — all simultaneously, without disrupting existing customers.

KEY PRODUCT DECISIONS

Strategic Choices That Shaped the Platform

Every change was made to solve a real customer problem. We wanted customers to spend less time working around product limitations and more time focusing on their business, while keeping the solution easy to operate and scale.

Customer Problem	Product Decision
Rigid onboarding flow	Configurable document selection per request
Deployment required for every change	Configuration-driven platform with no-code controls
One webhook format for all	Configurable webhook payloads per integration
Single failure strategy	Configurable failure handling and retry logic
Migration risk	Full backward compatibility with dual-flow support
Limited debugging visibility	Transaction lifecycle logging and error framework
Enterprise-specific requirements	Feature flags and request-level configuration

PLATFORM ENHANCEMENTS

What We Built

Enhancements were organized across four dimensions — each addressing a distinct layer of the enterprise experience, from core architecture to developer tooling.

Platform

- Modular architecture
- Transaction lifecycle redesign
- Zero-downtime migration

Developer Experience

- Enterprise configuration engine
- Feature flags & flexible APIs
- Improved documentation

Reliability

- Retry handling & thread-safe token refresh
- Partial failure handling
- Unified error framework

Enterprise Flexibility

- Configurable document fetching
- Webhook customization
- Domain whitelabeling

DEVELOPER EXPERIENCE

Designing for Developers

One of the biggest learnings from this project was that a great product isn't enough if it's difficult to integrate. Enterprise customers value products that are easy to implement and configure, so improving the integration experience became just as important as adding new capabilities.

"Reducing implementation effort is a product feature."

Flexible APIs

Introduced optional parameters and customer-specific configurations to simplify integrations and reduce repetitive implementation effort.

Configurable Webhooks

Customers define payload structure to match their systems.

Better Error Visibility

Clear transaction lifecycle logging accelerates debugging.

MIGRATION STRATEGY

Zero Downtime. Zero Risk.

Since the product was already live for enterprise customers, we wanted to avoid any changes on the client side. Instead of asking customers to migrate, we moved the platform to a new backend infrastructure while keeping existing APIs and integrations unchanged. This ensured a seamless transition with no impact on customer implementations.

Phase 1: API Compatibility

1

Migrated the backend while keeping customer-facing APIs unchanged, resulting in zero integration changes.

2

Phase 2: Feature Flags

Request-level flags enable gradual rollout to willing enterprise customers for specific features.

Phase 3: Database Migration

3

Migrated the entire legacy dataset to the new database with validation and rollback safeguards to ensure a seamless transition.

4

Phase 4: Full Rollout

Legacy flow was deprecated only after formally notifying customers and confirming a successful transition.

0

Downtime Events

100%

Backward Compatible

IMPACT & OWNERSHIP

Results, Ownership & Learnings

Product Impact

- **+7%** verification success rate improvement
- Higher enterprise adoption across clients (20K TO 60K+ hits a day)
- Multiple configurable enterprise workflows live
- Zero-downtime migration completed
- Improved developer experience and onboarding
- Scalable and fresh User experience

What I Owned

- Analyzed customer pain points and implementation gaps
- Prioritized roadmap and defined configurable capabilities
- Designed the enterprise configuration framework
- Led migration planning and rollout strategy
- Worked closely with engineering and QA throughout

Key Learnings

Products serving enterprise customers should be **configurable by design,** enabling different business needs without requiring custom development.

Developer experience directly influences enterprise adoption. If integration is hard, the product loses before it starts.

Backward compatibility reduces migration risk and builds customer trust during platform transitions.