

Unified Offers Hub: *Standardizing at Super-App Scale*

How fintech app managed to replace three siloed backend codebases with a single version-controlled contract, eliminating software releases for marketing initiatives across 1,8m MAU

3 Platforms, 3 Data Models, 3 Sprints per Offer

THE LEGACY REALITY

The Mobile, Web, and iBank teams didn't talk at the API layer.

Every promotion or cashback offer in the super-app was hardcoded differently. Mobile had its own JSON payload, Web maintained a separate microservice, and Internet Banking (iBank) duplicated logic in a third repository.

PRODUCTION FRICTION

2 to 3 Week Sprints: Launching a single promo category required 3 frontend deployments and coordinated backend releases.

Data Drift: A user seeing a 10% cashback offer on the Mobile App often saw outdated terms on Web due to decoupled DB syncs.

Commercial Bottleneck: Business teams couldn't run time-sensitive campaigns without stealing core engineering resources.

The 4-Month Confluence Spec War

ARCHITECTURAL ALIGNMENT

Contract First, Code Second

Before touching a single line of microservice code, I spent 4 months authoring the Master Specification space on Confluence. We fought over data schemas, backwards compatibility, and payload sizes until all three platform leads signed off.

SPEC GOVERNANCE

Key Spec Invariants

- 1. Single Payload Contract:** One JSON schema serves Mobile, Web, and iBank simultaneously.
- 2. Campaign Mode Integration:** Reused the exact same parametric schema as our real-time personalization engine.
- 3. Version Guardrails:** Older app clients gracefully fall back without throwing 500 API errors.

Zero Releases: Parametric Badge Schema

CONTRACT SPECIFICATION

How Commercial Teams Provision Offers Without Code

Instead of building bespoke UI components per offer, the frontend engines render UI components dynamically based on dictionary tokens and parametric data schemas.

When marketing wants a new Black Friday campaign, they insert a database entry mapping tokens to visual elements. Frontend apps render it immediately without an App Store update.

Velocity Shift: From 3 Weeks to <1 Hour

<1h

TIME-TO-MARKET (DOWN FROM 21 DAYS)

SCALE IN PRODUCTION

900,000 Unique Active Users in December Alone

During the December holiday campaign rush, commercial teams launched dynamic seasonal hubs, including Black Friday, New Year, and partner cashbacks, across the entire super-app ecosystem purely through database configuration.

Zero emergency patches. Zero frontend deployments. 100% data consistency across all digital touchpoints.

How 3 Teams Actually Migrated

PHASE 1

Mobile First

We prototyped and battle-tested the spec with the Mobile App team first. Their feedback led to refining version guardrails for legacy client support.

PHASE 2

Web & iBank Onboarding

Web and iBank teams reviewed the spec and proposed web-specific rendering adapters. We iterated on the Confluence Master Doc until all edge cases were covered.

PHASE 3

Legacy Sunset

Redirected database traffic flows to the unified gateway and systematically decommissioned services across all three infrastructure stacks.

Unintended Advantage – REFER A FRIEND

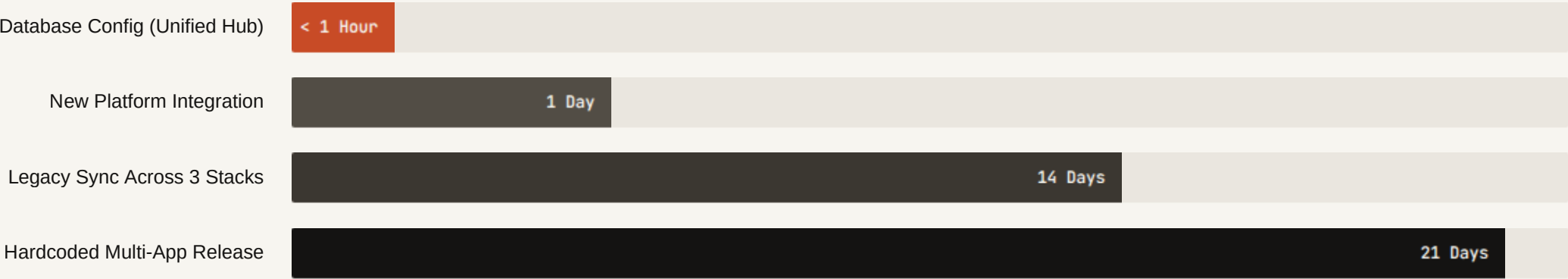
SYSTEM REUSABILITY

Zero New Microservices for a Full Referral Engine

When we wanted to launch a premium referral system for different segment customers in late 2025, we didn't build a new backend.

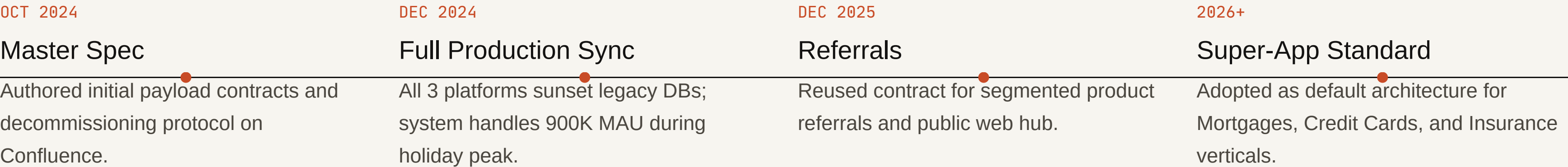
Instead, we plugged referral mechanics directly into the existing Offers Hub contract

The same API contract also powers our public web portal for unauthenticated visitors.



By removing frontend software dependencies, we gave commercial teams full agility while freeing core platform engineers to build infrastructure.

Standardization Timeline



What We Learned (The Hard Way)

LESSON 01

Spend Time on Docs, Not Fixes

4 months of spec debate saved us years of tech debt. When teams disagree on API contracts, solve it in Markdown/Confluence before writing microservices.

LESSON 02

Decouple UI from Deployments

If marketing needs a software sprint to change a banner or offer logic, your frontend and backend are tightly coupled in the wrong places.

LESSON 03

Standardize Payloads, Not Stacks

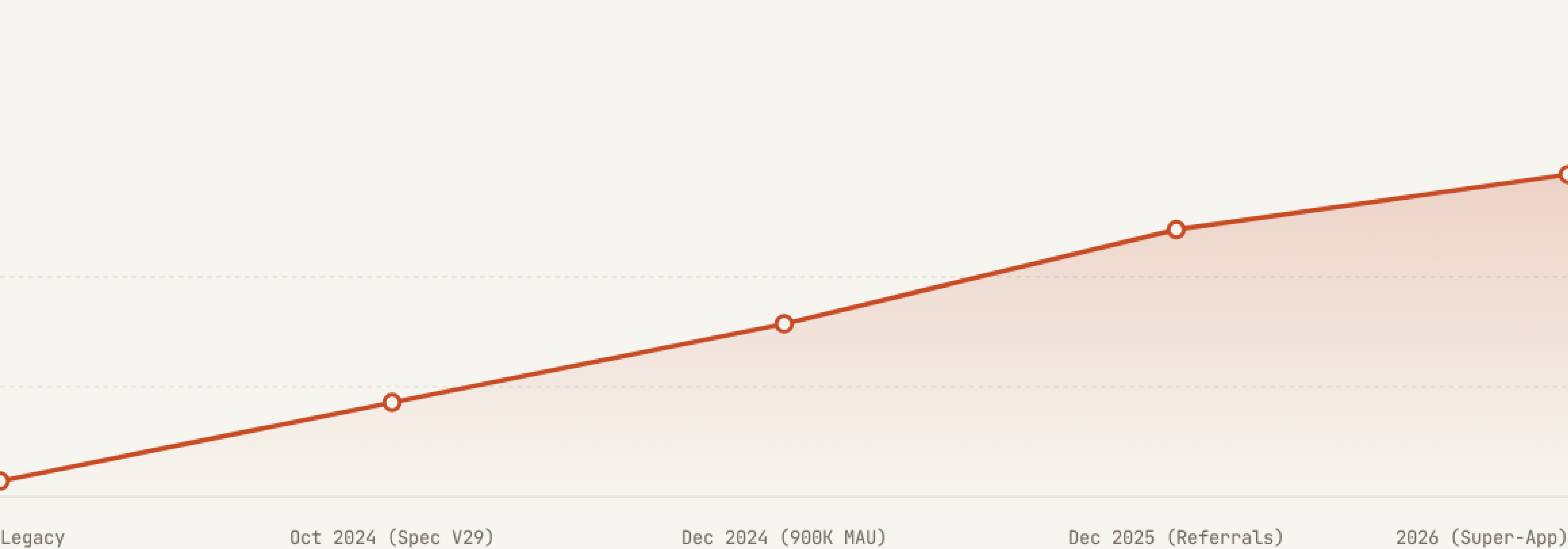
Mobile (Swift/Kotlin), Web (React), and iBank had different internal frameworks. We didn't force them onto one stack—we forced them onto one JSON payload contract.

LESSON 04

Build for Adjacent Use Cases

A well-designed contract for offers naturally scaled to handle referrals and public web pages without requiring bespoke backend rewrite projects.

Offers Hub Engagement Trajectory



The standardized contract supported rapid growth without degrading API performance or requiring emergency database scale-ups.

RAPTORS CONFERENCE 2026

Questions & Discussion

*"Real standardization isn't about imposing rigid frameworks on developers
it's about creating clean abstractions that let teams move fast without stepping on each other."*