CASE STUDY #2

Dual Payment Gateway Checkout Without Losing Orders Mid-Flow

Tatoo -- Checkout Resilience Across PayPal and Stripe

Project	Tatoo -- Tattoo and Sticker Marketplace
Area	Payment Processing, Checkout Flow, Order Integrity
Gateways	PayPal, Stripe
Difficulty	High -- Async gateway flows, session interruptions, financial integrity
Outcome	Zero lost orders on gateway interruption; consistent checkout state across both providers

Background

Tatoo supported two payment options at checkout: PayPal and Stripe. Some users preferred PayPal's familiar flow and buyer protection. Others wanted to pay directly with a card through Stripe without leaving the site. Offering both expanded revenue options without adding complexity for the user. The complexity landed entirely on the backend, specifically in handling the fundamentally different ways each gateway completes a transaction.

The Problem

Problem 1 -- PayPal Redirect Flow Created Abandoned Orders

PayPal standard checkout redirects the user away from the merchant site to PayPal pages to authenticate and approve payment. The user then returns via a redirect URL with a token confirming approval. The problem was what happened in between. If the user closed the PayPal tab, hit the back button, or lost their connection mid-redirect, the order record on the Tatoo side was left in a pending payment state with no way to resolve it automatically.

The Stuck Order Problem

During staging tests, roughly 1 in 8 PayPal checkout attempts resulted in a stuck order. The user had not been charged, but the order record existed and the cart was not cleared. If the user tried again, they created a second pending order. Admins had no automated way to clean these up.

Problem 2 -- Stripe Webhook Delays Caused Confirmation Gaps

Stripe confirms payment via a server-side webhook rather than relying on the client-side redirect alone. This is more reliable but introduced a timing gap. The user completed payment and was redirected to the confirmation page before the webhook had arrived and processed. The page had no order to confirm and showed an error. The order was fine, but the user saw a failure screen and sometimes attempted to repurchase.

Problem 3 -- Switching Gateways Mid-Checkout Left Orphaned Records

A user who started a PayPal checkout then switched to Stripe created a pending PayPal order record that never resolved. Completing the Stripe payment left two records for the same cart: one successful Stripe order and one abandoned PayPal order. Inventory was not double-deducted, but the orphaned records accumulated and made order history confusing for admins.

The Solution

1. Checkout Session With Explicit Lifecycle States

Rather than creating an order record at the start of checkout, a lightweight checkout session record was introduced. The session held cart contents, selected gateway, and a status moving through defined states: initiated, gateway-redirected, payment-confirmed, and order-created. An order record was only created after payment was confirmed. Abandoned sessions expired after 30 minutes and were cleaned up automatically with no orphaned order records left behind.

2. PayPal Token Polling for Interrupted Redirects

When a user returned to Tatoo after a PayPal redirect, the frontend immediately polled a status endpoint using the PayPal order token. If PayPal reported the payment as approved, the backend created the order and confirmed the session. If PayPal reported it as pending or cancelled, the session was reset and the user was returned to the checkout page with their cart intact and a clear message about what happened. No stuck orders, no empty cart.

3. Polling Bridge for Stripe Webhook Delay

On the Stripe confirmation redirect, instead of immediately rendering a success or failure page, the frontend entered a short polling loop against a lightweight order-status endpoint. The endpoint checked whether the webhook had arrived and the order was confirmed. If confirmed, the success page rendered. If not yet confirmed after 10 seconds of polling, the page showed a processing your payment holding state. The webhook almost always arrived within 2 to 3 seconds. Confirmed orders were never shown as failed again.

4. Gateway Switch Invalidation

When a user selected a different payment gateway than the one on their current checkout session, the previous session was explicitly invalidated before the new one was created. No new session started until the old one was marked as cancelled. This eliminated the orphaned record problem entirely. Admins saw clean order history with one record per completed transaction.

Problem	Solution
PayPal redirect abandonment creates stuck orders	Checkout session lifecycle; orders created only after confirmation
Stripe webhook delay shows false failure screen	Client-side polling bridge holds confirmation page until webhook lands
Switching gateways creates orphaned records	Explicit session invalidation before new gateway session starts
Abandoned sessions pollute order history	Session TTL with automatic expiry and cleanup after 30 minutes

Outcome

After the checkout session architecture shipped, stuck and orphaned orders disappeared from the system. Admins reported clean order history with no unexplained pending records. Users completing PayPal checkouts who interrupted the redirect were returned to a working cart rather than a broken state. Stripe confirmation pages showed success accurately without requiring a page refresh. Both gateways behaved consistently from the user perspective even though their underlying flows were completely different.

Key Takeaway

Multi-gateway checkout is not just a matter of connecting two different APIs. Each gateway has its own timing model, redirect behavior, and failure mode. Designing a checkout session layer that sits above both gateways and owns the order lifecycle is what makes a resilient system possible. The gateway is a detail. The session is the source of truth.

-- Ehsan