TESTMU AI · CUSTOMER CASE STUDY

Why a Payments Platform Stopped Trusting Retried Passes

60+ Tests found passing only on retry

9 of those masked genuine product race conditions

3x Faster failure triage once every failure was classified by cause

About the Company

The company is a Series C payments platform that processes card and bank transfer payments for online merchants, sitting in the settlement path between shoppers and the businesses they buy from. Its 150 person engineering organization, within a company of roughly 400, owns everything from checkout integrations to the reconciliation logic that runs behind them.

That scope raises the cost of a defect. A race condition in this codebase doesn't produce a display error; it can duplicate a charge, and a bug that reaches production means money moving incorrectly for a merchant and their customers. That exposure is what turned a routine effort to reduce flaky tests into a review of what a passing test actually establishes.

The Challenge

Eighteen months earlier, the team had set `retries: 2` in its Playwright configuration. The reasoning was sound then. Flaky tests were producing enough red builds to obscure genuine failures, and retries removed the noise.

The effect was different from the intent. A test could fail on the first attempt, pass on the second, and the pipeline would report a green build. The report showed a pass, so nobody reviewed it further.

A genuine race condition in the checkout flow followed that same path. It failed once, passed on retry, shipped, and surfaced in production during a merchant’s peak sales period as intermittently duplicated payment attempts. The postmortem identified the test that had detected it and established that the test had been detecting it for weeks. The retry suppressed the result each time.

The same postmortem showed how limited triage had become. When a run failed, the report did not indicate whether the cause was the application, a stale locator, outdated test data, or a dropped browser session. All four presented identically as a red result, so the team re-ran the suite rather than investigating.

The reverse case was equally difficult. The suite could not detect an action that had silently done nothing. An automated step would click, the application would not respond, the next step would not verify the outcome, and the run would complete green. For a payments product, neither blind spot was acceptable.

Why TestMu AI

The engineering director's requirement after the incident was that no green build should go unexamined.

The team adopted Kane CLI because its execution evidence makes each run inspectable: what every step did to the application, why a failure occurred, and whether a pass was earned or achieved on a second attempt. Under that model, a pass on retry is no longer simply a pass. It becomes a signal in its own right, with the evidence from the first failing attempt retained rather than overwritten.

The Solution

Kane's evidence-first execution changed what a red and a green result each mean in practice.

Failures now arrive already classified from their artifacts, distinguishing application behavior that diverges from the acceptance criterion, a test-side defect such as a stale locator, and an infrastructure fault such as a dropped session. Engineers no longer open several tools to work out which category a failure belongs to.

Retried passes are flagged with the evidence from both attempts shown together, which turns the existing retry configuration into a means of identifying flaky tests rather than hiding them.

Step-level verification addressed the no-op gap. Kane confirms that each action produced its expected effect on the application before counting the step as complete, so an action that changes nothing fails at the point it occurs rather than passing through to the end of the run.

The Results

The first full audit of retried passes identified more than sixty tests that only passed on a second attempt. Nine were masking genuine product race conditions of the same type as the one that had reached production. All nine were resolved before they affected customers.

Triage time fell by roughly two-thirds, because each failure now arrives with its cause attached. The team has retired re-running a suite as a diagnostic step.

"We configured the retries ourselves to reduce the noise in our build channel. It took a production incident to see that we had also removed our ability to detect the failures that mattered."

Director of Engineering, payments platform

"A red result used to mean the cause could be anything. It now tells me something specific before I open the report."

SDET, payments platform

Ready to improve your testing? Schedule a demo to see how TestMu AI can help your team achieve similar results.

Industry: Fintech, Payments

Company profile: Series C payments platform

Employee size: 400+ employees

Product: Kane CLI