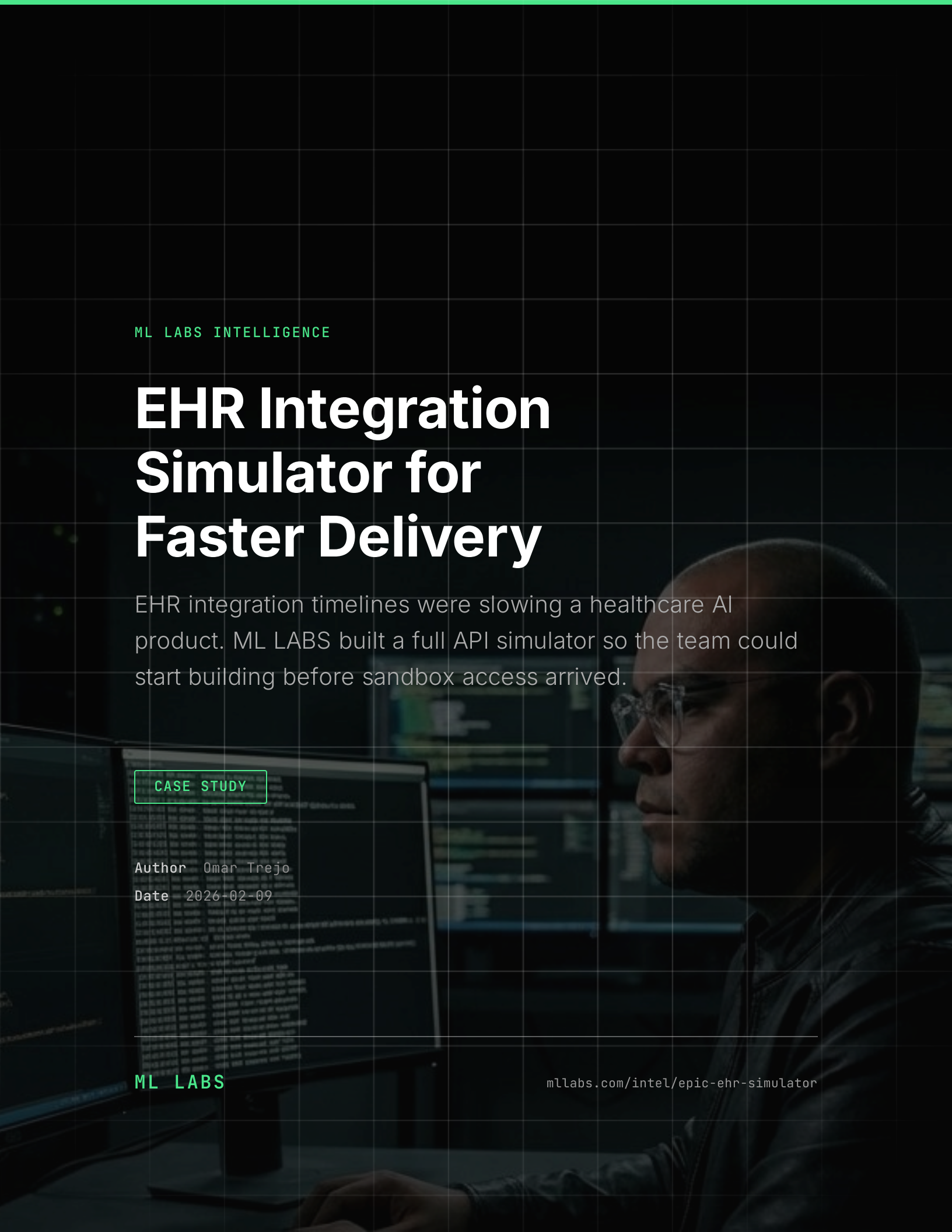
ML LABS INTELLIGENCE

EHR Integration Simulator for Faster Delivery

EHR integration timelines were slowing a healthcare AI product. ML LABS built a full API simulator so the team could start building before sandbox access arrived.



CASE STUDY

Author Omar Trejo

Date 2026-02-09

HeartSciences needed to integrate their AI-ECG platform with a major EHR vendor's electronic healthcare record system. EHR integrations are notoriously slow — sandbox access is limited, environments are shared, and every test cycle depends on external coordination. The integration required bidirectional HL7 messaging (order and result messages) alongside FHIR OAuth flows, each with per-hospital configuration differences that only surface during live testing.

ML LABS built a full EHR API simulator that replicated the targeted workflows and API surface at the protocol level, letting the team develop and test HL7 message handling, FHIR session management, and per-organization field mappings immediately — before sandbox access was available.

The Problem

EHR integration is one of the hardest operational bottlenecks in health-tech. The constraints compound: sandbox environments are shared and often unavailable when needed, each test cycle requires coordination with the vendor's support teams, and integration bugs found late in the process trigger expensive rework. But the real complexity goes deeper than access scheduling.

The integration surface for an AI-ECG platform includes unsolicited order workflows (order messages that the EHR sends without a prior request), result delivery with diagnosis segments that affect billing, and FHIR-based clinical launch contexts with organization-specific OAuth binding. Each hospital configures these flows differently — field mappings vary, optional segments become required, and ordering provider identification shifts between segments depending on the site. Discovering these variations against a live sandbox is slow and expensive. Discovering them in production is unacceptable.

What the Simulator Replicated

The simulator wasn't a generic FHIR mock server. It replicated the specific integration behaviors that the AI-ECG platform depended on — the ones that would have required waiting on vendor sandbox access to develop and test against.

HL7 Message Workflows

The core clinical workflow is unsolicited: the EHR sends an order message when a physician requests an ECG interpretation, and the platform must respond with result messages containing the AI analysis. The simulator replicated this bidirectional flow including per-org message behavior — some hospital configurations expect the platform to initiate unsolicited results, others expect results only in response to explicit orders. The simulator exercised both paths.

The simulator generated and validated against observation segments carrying RR intervals, P/R/T axis values, and waveform measurements; diagnosis segments with billing codes that downstream revenue cycle systems depend on; and placer order number tracking through fields that link results back to the originating order. The simulator also handled multiple diagnosis segment generation, which matters when a single ECG study produces findings across several diagnostic categories.

FHIR Authentication and Security Boundaries

The vendor's FHIR integration requires per-organization authentication — each hospital system authenticates through its own endpoint with credentials tied to that organization. The simulator replicated these per-organization security boundaries, including adversarial scenarios where requests attempt to cross organizational boundaries. This verified that the platform's tenant isolation held under conditions that would only surface in multi-organization production deployments.

> The simulator's value wasn't in replicating the vendor's API surface — it was in replicating the per-organization behavioral differences that only surface during live integration testing.

Per-Organization Configuration

Different hospitals need different HL7 field mappings. Patient identifier formatting varies by site. Ordering provider representation differs between organizations. The simulator maintained per-organization configuration profiles that mirrored the variation the platform would encounter in production, allowing the team to develop and test configuration-driven parsing logic without access to each hospital's actual EHR instance.

How the Simulator Accelerated Delivery

The simulator removed the external dependency from the development cycle, but its impact went beyond schedule compression. It changed the discovery pattern for integration bugs — surfacing issues during development rather than during limited sandbox windows.

The return order generation logic for the unsolicited workflow was developed entirely against the simulator. Without it, this flow would have required coordinated testing sessions with the vendor's team for each iteration. The simulator allowed rapid iteration on message handling and response generation without external dependencies.

Patient identifier and ordering provider field corrections were discovered through simulator testing. The original parsing logic made assumptions about field formatting that held for the first hospital configuration but broke for subsequent ones. The simulator's per-organization profiles exposed these assumptions before they reached a live environment, saving rework cycles that would have been measured in weeks of sandbox coordination.

Test Infrastructure

The simulator also served as the foundation for the platform's integration test suite. Unit tests covered message handling across all supported organization configurations. End-to-end tests exercised the full clinical workflow from order receipt through AI processing to result delivery, verifying correct output for each organization profile.

This test infrastructure remained valuable long after initial development — it served as the regression safety net through production launch and subsequent feature additions.

Boundary Conditions

This approach works when the integration surface is well-defined and the behavioral variations are discoverable. The vendor's HL7 and FHIR interfaces are documented (if incompletely), and the per-organization variations follow patterns that can be catalogued. The simulator is not a substitute for real EHR validation — it is a development accelerator that compresses the iteration cycle so that real sandbox time is spent on validation rather than discovery.

For integrations where the API surface is undocumented, actively changing, or where the behavioral variations are not pattern-based, a simulator provides less value. The effort to build an accurate simulator exceeds the effort to work directly against the real system. The key question is whether the external dependency is a scheduling bottleneck or a knowledge bottleneck — simulators solve the former, not the latter.

First Steps

If your team is blocked by external system dependencies — whether an EHR vendor or any third-party API with limited test access — the pattern is the same.

1. **Build a simulator that covers your specific integration surface, not the entire API.** Start with the workflows that carry the most risk and the most per-organization variation. For EHR integrations, that typically means the order-result messaging flow and the authentication binding.
2. **Use real system interactions to calibrate the simulator.** Every sandbox session should produce updated simulator profiles that capture newly discovered behavioral differences. The simulator gets more accurate over time rather than drifting.
3. **Invest in the test infrastructure around the simulator.** The simulator's long-term value is as a regression testing foundation, not just a development stand-in. Build the test suite as you build the simulator.

Practical Solution Pattern

Simulate the specific integration behaviors that block development — per-organization field mappings, bidirectional message workflows, and authentication binding — rather than waiting for external sandbox access for each iteration cycle. Calibrate the simulator against real system interactions so it captures behavioral variations that documentation misses, and build a regression test suite on top of it that serves as the safety net through production launch and beyond.

This works because the bottleneck in EHR integration is rarely engineering complexity — it is the iteration speed imposed by external dependencies. Integration development that typically requires months of sandbox coordination was compressed to days — over 10x faster time-to-first-integration compared to the standard EHR vendor onboarding timeline. A calibrated simulator compresses what would be weeks of coordinated sandbox sessions into hours of local development, and the test infrastructure it enables catches regression issues that would otherwise surface during production validation. Pattern recognition from prior EHR integrations compresses what would otherwise be an expensive learning curve — the difference between a first attempt and an experienced execution is not incremental, it is categorical. If one healthcare integration workflow is already defined and blocked by technical dependencies, **AI Workflow Integration** is the direct build path.

References

1. **Open EHR FHIR Documentation**. 2024.
2. HL7 International. **FHIR Standard**. *HL7*, 2024.
3. HL7 International. **HL7 v2.x Messaging Standard**. *HL7*, 2024.

4. ONC. **Interoperability Standards Advisory**. *Office of the National Coordinator for Health IT*, 2024.
5. SMART Health IT. **SMART on FHIR Authorization**. *HL7/Boston Children's Hospital*, 2024.