HYPERTEST

Make Integration Testing easy for Developers & Agile Teams

Discover proven strategies to eliminate integration failures in your apps & services

WHITEPAPER

TABLE OF CONTENTS

What Makes Integration Testing Difficult for Developers?

> **The hidden cost of microservices is managing integration failures**

All modern applications are distributed. This design helps accelerate development and maintain performance, reliability and fault tolerance at scale.

But this exposes modern applications to the risk of integration failures i.e. issues and downtimes because of failing interactions between your code and its dependencies.

Consider incorrect database calls, failing inter-service or 3rd party API contracts and erroneous async operations.

Achilles Heel for all modern distributed backends

01

DOWNSTREAM CHANGES ➡ UPSTREAM FAILURES

Frequent service changes lead to unexpected upstream failures. This becomes even harder to predict with multiple dependencies or unclear dependency chains, causing chaos for developers

02

WRITING INTEGRATION TESTS HARDER THAN DEVELOPMENT

2 problems: Devs mostly write unit tests, which are never enough. Second, testing integrations means simulating or mocking dependencies which is never easy to get right

03

COLLABORATION IN MULTI-SERVICE SETUP NOT EASY

Very hard for devs to collaborate in a large team over changes that involve multiple services. What it gives? Larger teams and too many services

04

DEBUGGING NIGHTMARE

Debugging these issues is difficult for developers. Tracing the failing request through multiple services lacks any easy way. Most devs struggle to pinpoint the origin of the problem

Why Unit Tests are NOT enough?

Unit tests are useful for checking the logic within a service but fail to test the dependencies between services.

Too much reliance on unit tests has its risks:

UNIT TESTS HAVE LIMITED SCOPE

Microservices interact with databases, services, queues, and APIs. Unit tests focus on isolated code fragments but not these interactions. For services, a broad testing scope is crucial.

HIGH COVERAGE ➡ LOW QUALITY

Dev writes a test for adding numbers, but only checks if the output is a number, not the correct sum. This low-quality test covers the function but misses a bug.

INCOMPLETE TESTING

Unit tests miss interaction failures. These, often the trickiest issues, can cause data loss, slowdowns, or production crashes.

NO COVERAGE FOR VALIDATIONS

Validations check logical boundaries but tests that cover all the lines but poorly validate these boundaries still end up reporting high coverage.

Integration Testing made easy for Developers with HyperTest

> HyperTest is a game changer for us in integration testing, It has significantly saved time and effort by green-lighting changes before they go live with our weekly releases.

Vinay Jaasti
CTO, Airmeet

HyperTest captures real interactions between code and external components using actual application traffic, then converted into integration tests

TESTS INTEGRATION SCENARIOS

It verifies data and contracts across all database, 3rd party API calls and events

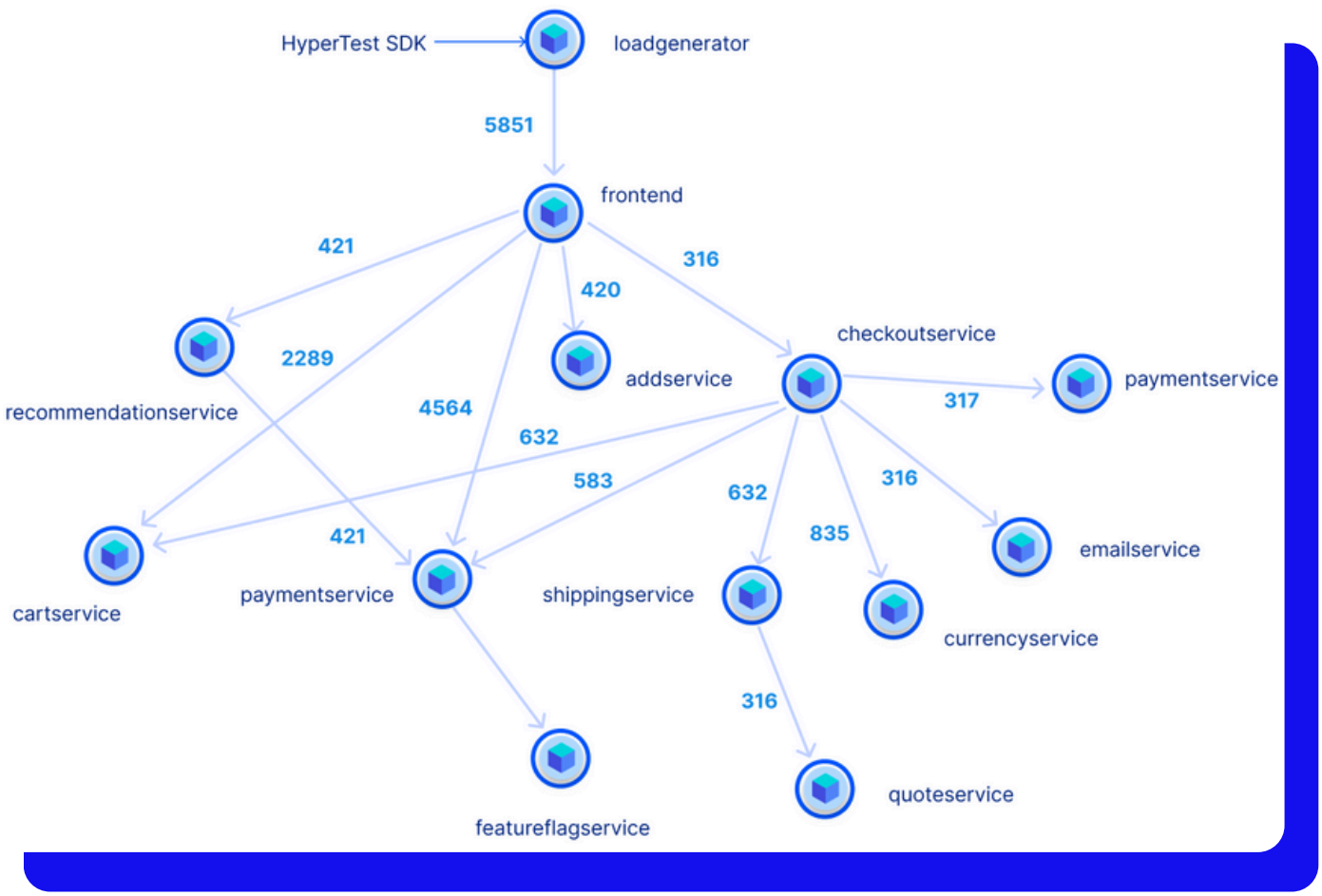
SMART MOCKING

HyperTest mocks external components and auto-refreshes mocks when dependencies change behaviour

RUN WITH CI OR LOCAL

These tests can be run locally or with CI pipeline much like unit tests

1 Observability



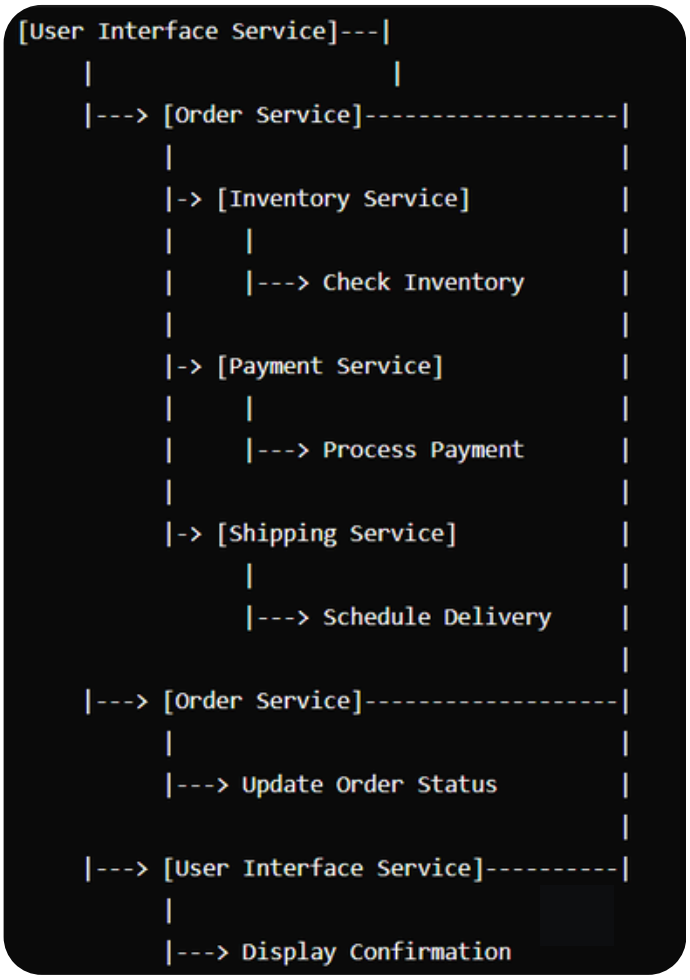
HyperTest is initialized on every microservice with its SDK.

- It then generates the trace of every incoming call i.e. request, response, outgoing call and outbound response.

- When done for all services, it generates an observability chart that reports all upstream-downstream pairs i.e. relationship between all services.

2 Distributed Tracing

HyperTest ***context propagation*** provides traces that spans multiple micro-services and helps developers debug the root cause of any failure in a single view



3 Support Multiple Protocols

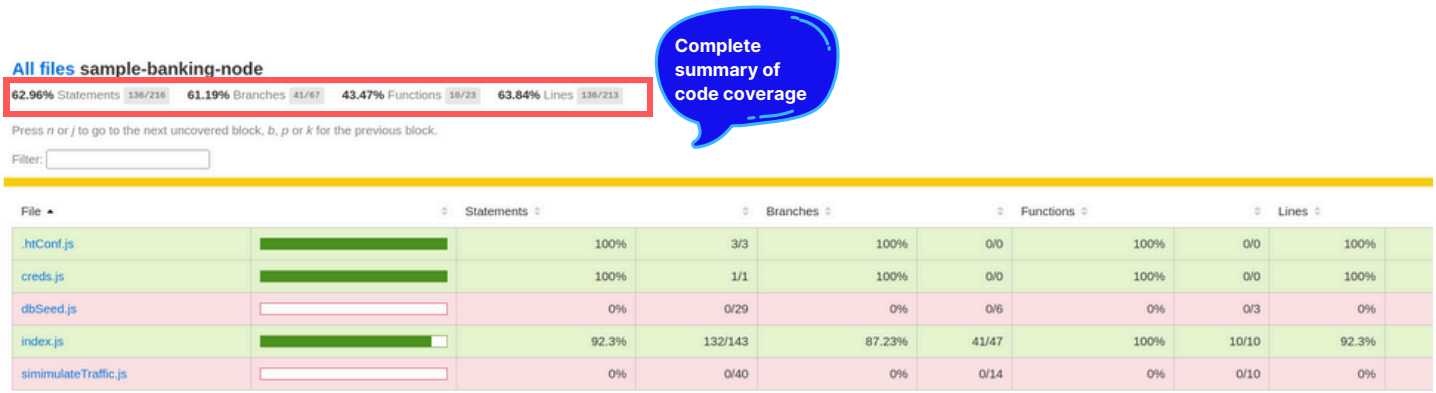
HyperTest is capable of supporting all the commonly used web protocols like HTTP HTTP 1.1 , HTTP 2 LIKE GraphQL, gRPC etc.

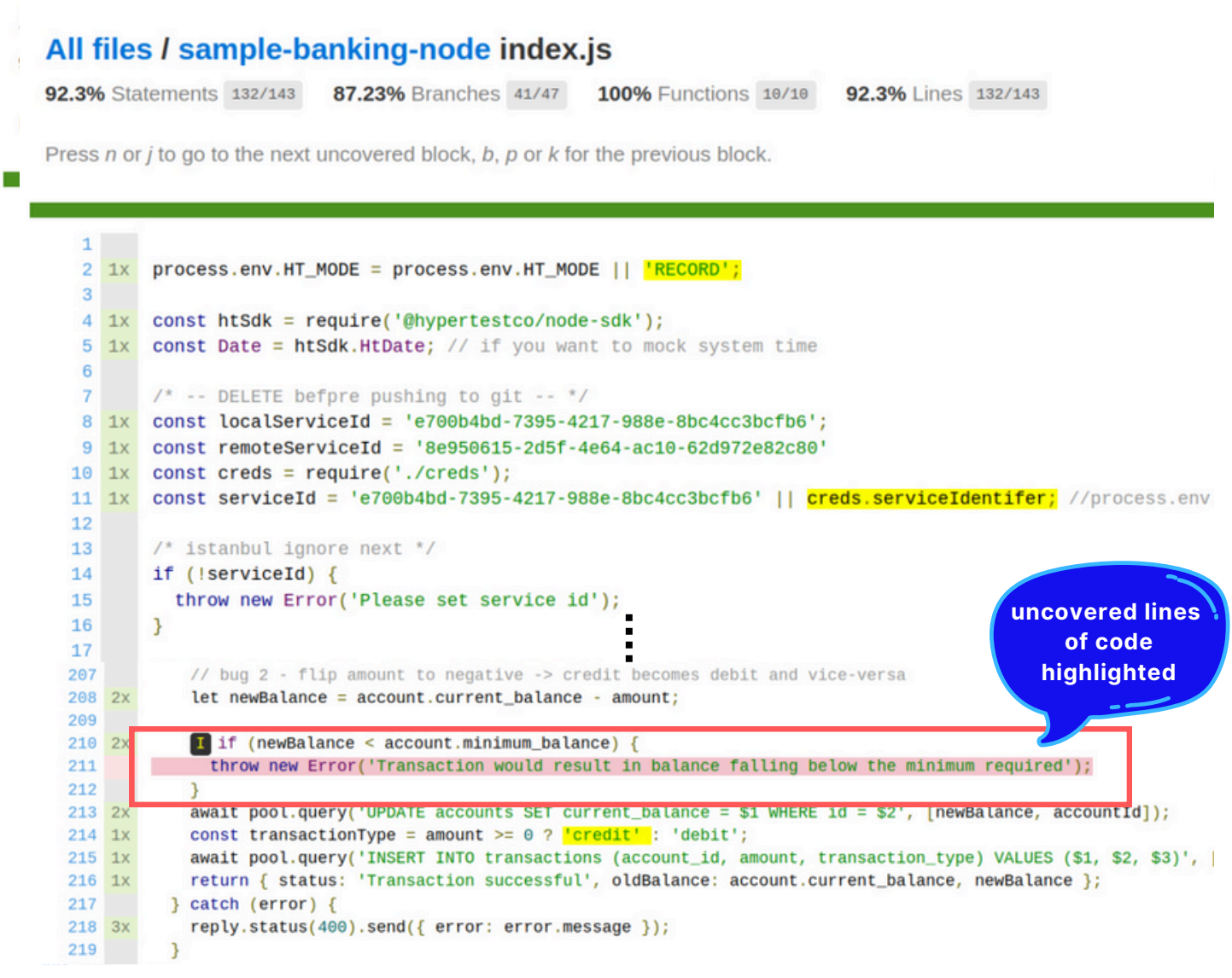
Also supports all non-http calls like databases, queues like Kafka, NATS, RabbitMQ and all pub/sub systems

- **HTTP, HTTP1.1, HTTP 2**
- **SQL / noSQL databases**
- **Message Queues and Pub/Sub Systems**
- **Web sockets**

4 Code Coverage

HyperTest generates a **code coverage report** after every run. This highlights clearly how it tests code paths for both the data and integration layer along with the core logic



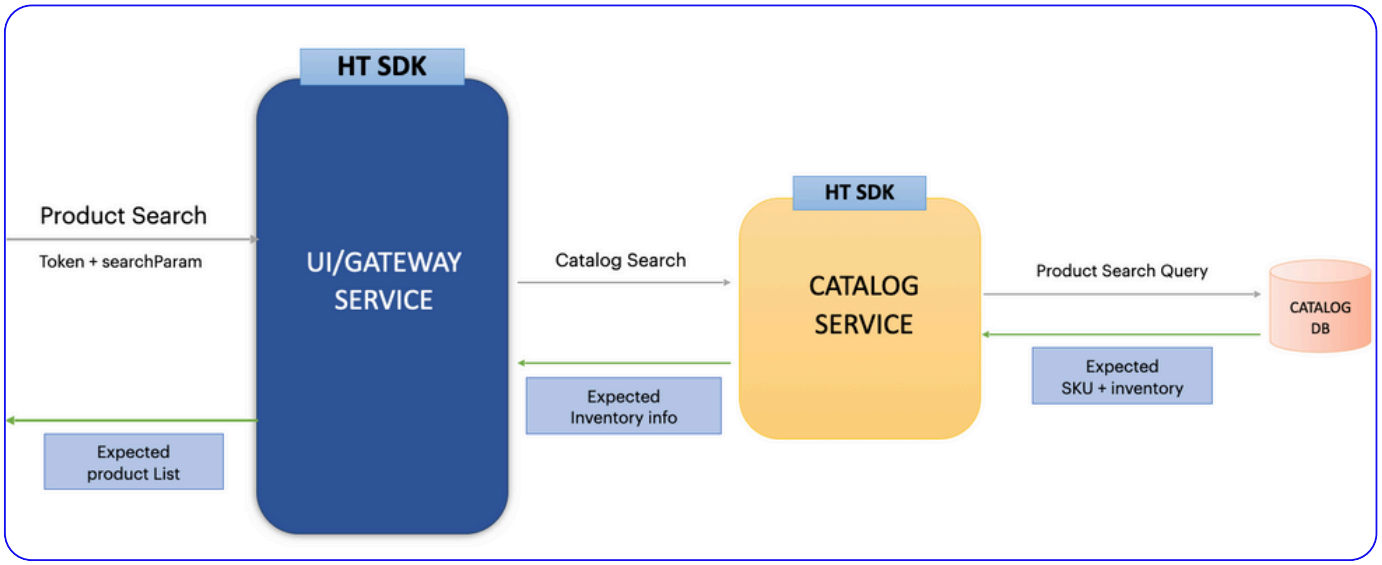


5 Smart-Mocking

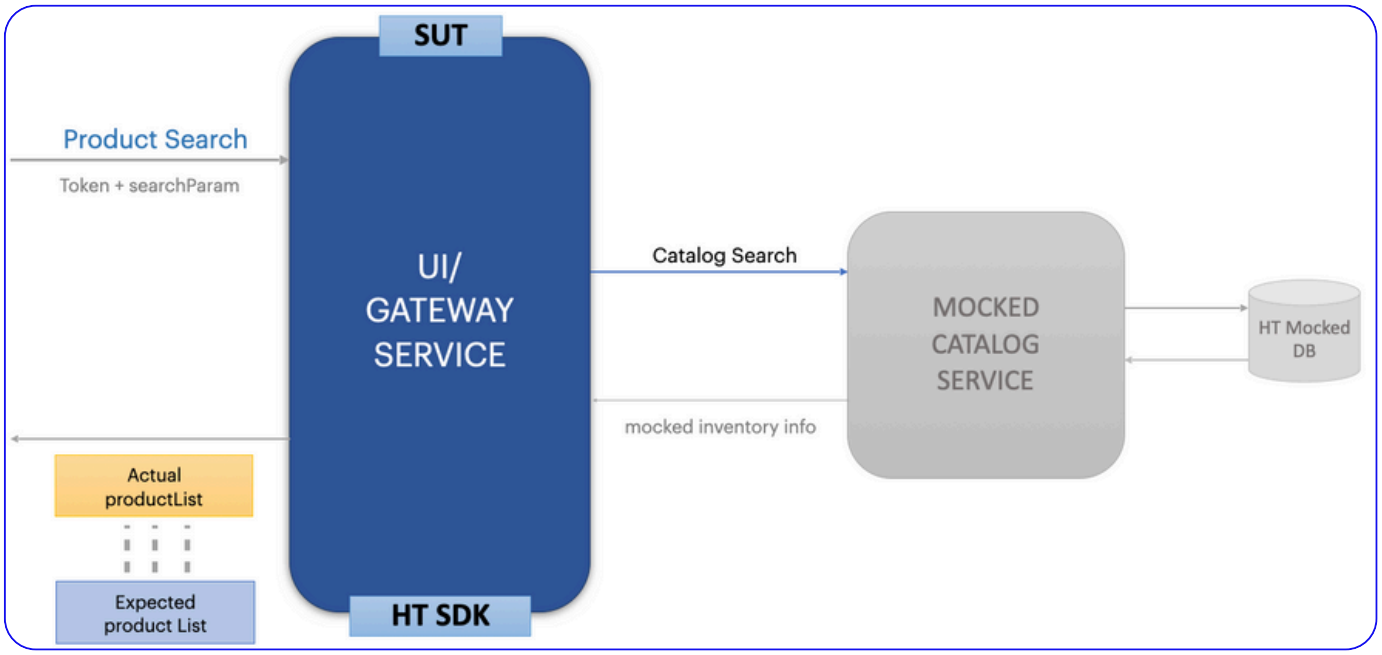
HyperTest smartly mocks external systems like **databases, queues, downstream or 3rd party APIs** that your code interacts with.

It also smartly auto-refreshes these mocks as dependencies change their behavior keeping tests non-flaky, deterministic, trustworthy and consistent

TEST GENERATION MODE

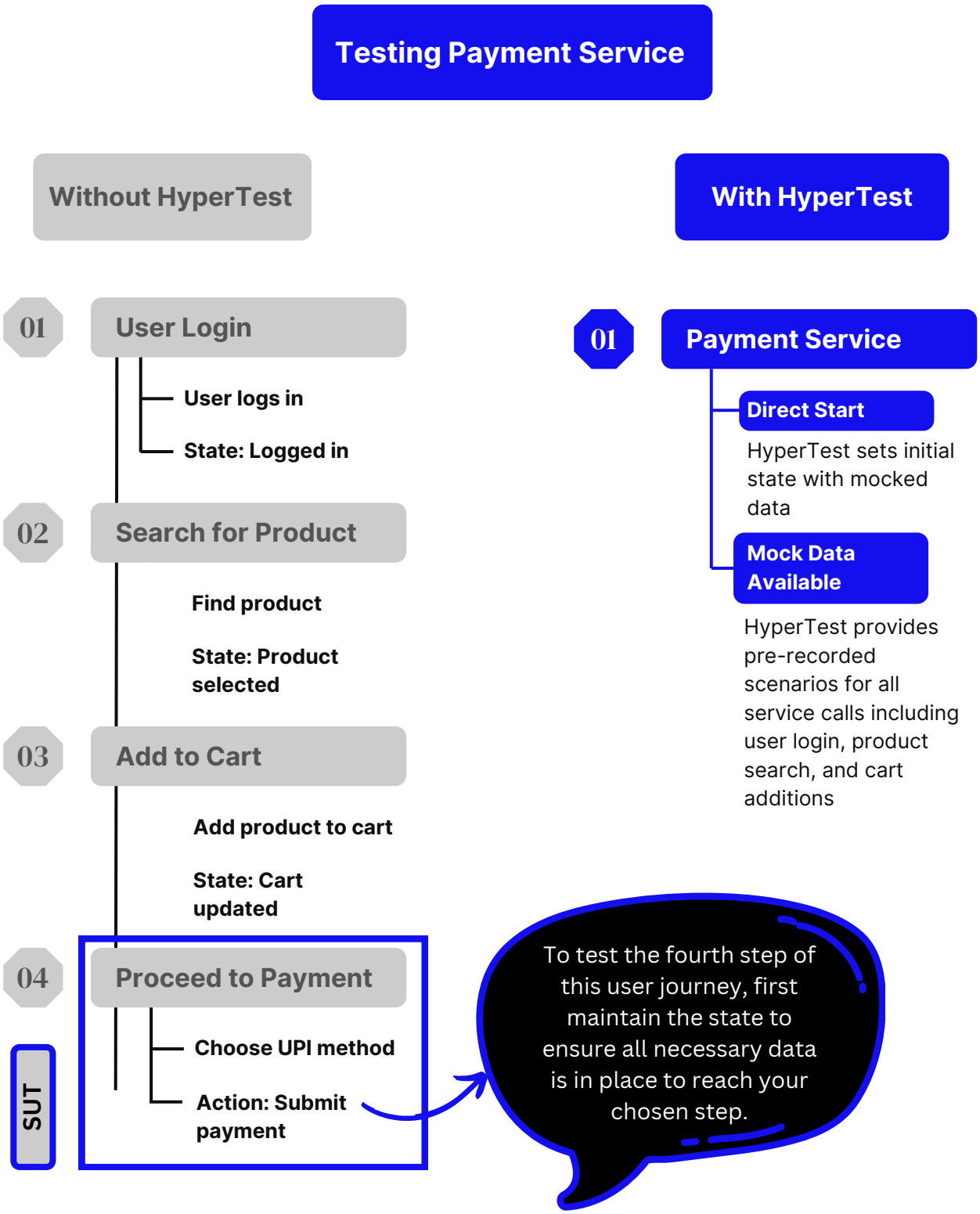


TEST MODE



6 Never worry about creating or managing test data

HyperTest can test stateful flows without needing teams to create or manage test data.



Summary

The hidden cost of microservices is integration failures.

- With the help of its capabilities like Smart mocking, no test data preparation, tracing etc., HyperTest is trying to make this uncertainty go away by making integration testing easy for developers. Teams should never leak integration failures.

2022

HyperTest was started

2023

Node SDK Launch

2023

100 services go live

2024

Launched our Java SDK

Generating Greatness

Companies **like Porter, Paysense, Nykaa, Mobisy, Skuad and Fyers** leverage HyperTest to accelerate time to market, reduce delays and improve code quality without needing to write or maintain automation