HYPERTEST

The CTO's guide to building an Autonomous API testing suite

WHITEPAPER

TABLE OF CONTENTS

EXECUTIVE SUMMARY

APIs carry the majority of any application's functional and business logic, and the pace at which they undergo change and are added, it becomes an avoidable task to test and protect them against failures and vulnerabilities. Sadly, building your own API automation suite is very time consuming. More often than not, returns suboptimal quality outcomes for teams.

In a fast paced development environment, it is often impossible to build your own API automation suite to test and secure APIs against breaking changes. Let's look at some of those challenges that every team faces and how the HyperTest platform mitigates that.

CHALLENGES	HYPERTEST
Keeping API documentation updated at all times	Discovers your APIs and documents for every schema currently under use helping inventory the API and generate a swagger
Writing an integration test for every new feature	Build a test case by monitoring how a new feature is tested in any lower prod environment. This is available as a regression test from the next sprint
Maintaining and updating all test cases to accommodate for the new changes	All auto-generated test cases automatically update themselves as the new build moves to production and updates the current baseline (stable version)

CHALLENGES	HYPERTEST
Thinking and writing assertions to cover every possible business rule that can break	We compare your version under test to the stable version to programmatically generate assertions. These assert every business rule deep and wide in your app without any input or prior knowledge
Continuously work towards improving the test coverage for your API suite	Use network traffic to continuously discover new flows, eliminating the effort of writing tests. Give teams more time to explore and expose more scenarios to HyperTest
Debug the error in minutes and understand the user context of the issue	Generates test cases as user journeys by monitoring network traffic. The error is reported across any step in the journey, thus makes it super easy to reproduce & debug
Catching API errors as close to development as possible	Integrates with the release process by triggering a test, and results as soon as a new commit or PR is available through the CI tool

OVERVIEW

Building your own API-automation suite can be very hard and time consuming. Even if we don't consider the investment in resources and manpower, the ability to deliver a suite that can catch any and every critical error pre-release is something most teams struggle to guarantee.

The reason behind these challenges is simple. Writing and maintaining test cases that can cover **every** work-flow is super hard. Even if teams hypothetically get somewhere, writing assertions that cover every business rule becomes another challenge.

HyperTest has looked to solve both challenges with a completely first principles approach that:

- Uses network traffic to automatically generate tests cases that also update themselves thereby eliminating the need to write any tests on your own

- Uses a simple test design principle, where the version under test is compared with the stable version, at the same time, to generate all assertions dynamically, removing the need to think or write assertions at all

BUILD AN AUTONOMOUS API-TESTING SUITE THAT NEEDS NO MAINTENANCE

Keeping an updated API inventory or documentation at all times, in a rapidly changing environment where new APIs are added all the time

CHALLENGE

Accuracy of documentation such as OAS is critical to help teams understand and assess the risk of APIs & exposure of data, and ensure they are aware of all the known and unknown usage patterns of their API schema

There is often a lot of gap between a manually created OAS and what's really deployed (in terms of the number of APIs documented and the level of detail captured in the documentation) vs. what HyperTest inventories when it discovers the actual API footprint from the application.

HYPERTEST

HyperTest scans the existing OAS or swagger document of the app and then compares it with the APIs it discovers from the actual traffic

This comparison gives us the list of those APIs that are shadow APIs and currently not in use and can be deprecated. This keeps the API inventory for the application up-to-date without maintaining an OAS file

Writing an integration test for every new feature that keeps the test sprint lag from the development sprint by one sprint

CHALLENGE

Test teams spend almost all their time in a sprint checking if any existing feature might break due to a new feature or change. For large, complicated applications, regression testing takes up almost the entire time and leaves little to no time to write automation for new features.

Test teams dedicate an entire sprint, after the development sprint is over to build automation for new features. This sprint size invariably limits the number of new story-points devs can complete in the dev sprint. This is because the regression suite that should ideally test all new features of the last sprint might still be not ready if it has to cover too many new features or story-points.

HYPERTEST

HyperTest can be set-up in any lower-prod environment to monitor user activity, be it stage, testing or even the local dev machine.

Before pushing a new feature to production, teams (QA or dev) do basic manual testing or sanity for the feature to see if it is working as expected.

This activity is usually done in any lower prod environment. These manual verification steps as traffic is enough for HyperTest to record and generate a test case for the new feature. Saves the time for the test team to write a test script for the same very steps he/she performed manually testing the feature.

Saves time and effort writing automation that needs an entire sprint!

Writing an integration test case for APIs is only half the job, those tests need to be also continuously updated to accommodate for the new changes

CHALLENGE

Any team that writes its own API integrations tests would know they also need to update these tests every sprint. The frequency with which new changes happen in the application dictate the pace at which such tests also need to be updated.

In fast paced agile set-ups, it often becomes difficult for engineering teams to convey the entire list of changes that are planned for the upcoming sprint, which results in incomplete updates to test suites, leaking errors. Building PRDs or detailed change-log documents slows down teams

HYPERTEST

HyperTest can be set up in any environment to monitor actual user actions and automatically generate test cases. It also uses the stable version of the application as a baseline or source of truth and compares its results to the test version for every test in real-time.

This means that all new changes reported as regressions by HyperTest in the first time for the current build (under test), get automatically updated as this build moves to production and becomes the new baseline. Therefor, teams don't have to update any test case separately.

Due to this design, all test cases are updated by themselves and need zero maintenance or input for update!

See Annexure A to understand the kind of integration test case generated by HyperTest as an actual user journey

---

Thinking and writing assertions to cover every possible business rule that can break the application in production

CHALLENGE

The second big challenge around keeping an API test suite effective is writing assertions that cover every business rule.

Most test teams write assertions on API responses that only cover basic regressions like status code errors, as logical problems are hard to write for, and hence the most difficult to catch. As new APIs are added, protecting newly added business rules with a system that relies on manually writing every assertion is hard to trust and scale.

This is why, despite the best efforts of test teams errors keep leaking into production.

Case in point: Imagine a loan processing app that calculates the EMI for the borrower. A bug in the calculation logic would process a loan with a much smaller EMI than expected. This logical is very hard to write an assertion for.

HYPERTEST

HyperTest has the most unique test design scheme that makes it impossible for devs to introduce a breaking change in their code that does not get reported. HyperTest by default compares the test build with the stable version and generates all assertions dynamically. The kind of logical errors HyperTest catches are very hard to write assertions for, explained in detail in Annexure B.

The comparison algorithm intelligently filters noise or removes false positives i.e. those changes that do not have potential to become a big error. The HyperTest algorithm automatically asserts every business rule in the application without qualification or manual input.

Annexure B explains how HyperTest intelligently asserts that the key time-stamp is a required key in every API response but its value can keep changing and is ideally not an issue.

Work continuously towards increasing the coverage of your API suite, maintaining an updated OAS file or swagger doc for all your services

CHALLENGE

The world is moving towards microservices and teams need updated OAS / swagger docs for every one of them to have the most accurate picture on the coverage of their API suite, and therefore the effectiveness of their automation suite.

There is always a large gap (sometimes to the extent of 50%) in manually created OAS / swagger doc maintained by the teams and what is really deployed, in terms of number of APIs documented and the level of detail captured by it.

In absence of an updated OAS / swagger doc that matches reality, teams struggle to track if their API automation suite cover rapidly added new APIs and old.

HYPERTEST

HyperTest ingests the existing swagger docs or OAS files of the application and analyses them to build a complete inventory of API endpoints, along with parameters and definitions.

When it monitors real traffic, it discovers all the API schemas that are exposed by actual traffic, tracking all possible combinations of schema scenarios that are actively used. What it does is help teams get a completely accurate and quantifiable sense of the coverage that network-generated API-tests from HyperTest provide.

Moreover, HyperTest helps teams that do not have OAS / swagger docs automatically generate one in minutes. HyperTest then continuously suggests addition or deprecation of new APIs based on its created baseline, and keeps the API inventory upto date automatically.

Annexure C explains how HyperTest discovers and reports API schemas along with parameters and how many times they are used across the app. Reporting API coverage then becomes conveniently simple.

In a DevOps world, all release are planned and delivered via CI /CD. Teams would like to release as and when required and with the absolute minimum number of issues

CHALLENGE

Agile teams would like to push new changes to production as many times as possible. The biggest hurdle becomes the absence of complete automation on their release process.

Also, with the help of a CI / CD enabled release process, teams can catch bugs and issues early in the release cycle. This might give devs enough time to fix problems without slowing down development.

HYPERTEST

HyperTest integrates with all popular CI tools and triggers tests natively from the CI pipeline. The build is failed or approved automatically inside the pipeline based on results.

HyperTest typically works with the release cycle of any team in the following manner:

1. The CI pipeline is triggered whenever a new merge request is created or a commit is made in that merge request
2. In the next step, the pipeline job starts a test on HyperTest by making an API call
3. After a test is run, it needs to be signed-off by a team member from HyperTest's regression report.
4. Once the new merge gets approved or rejected, the CI pipeline will continue. Else, it will be in waiting state.

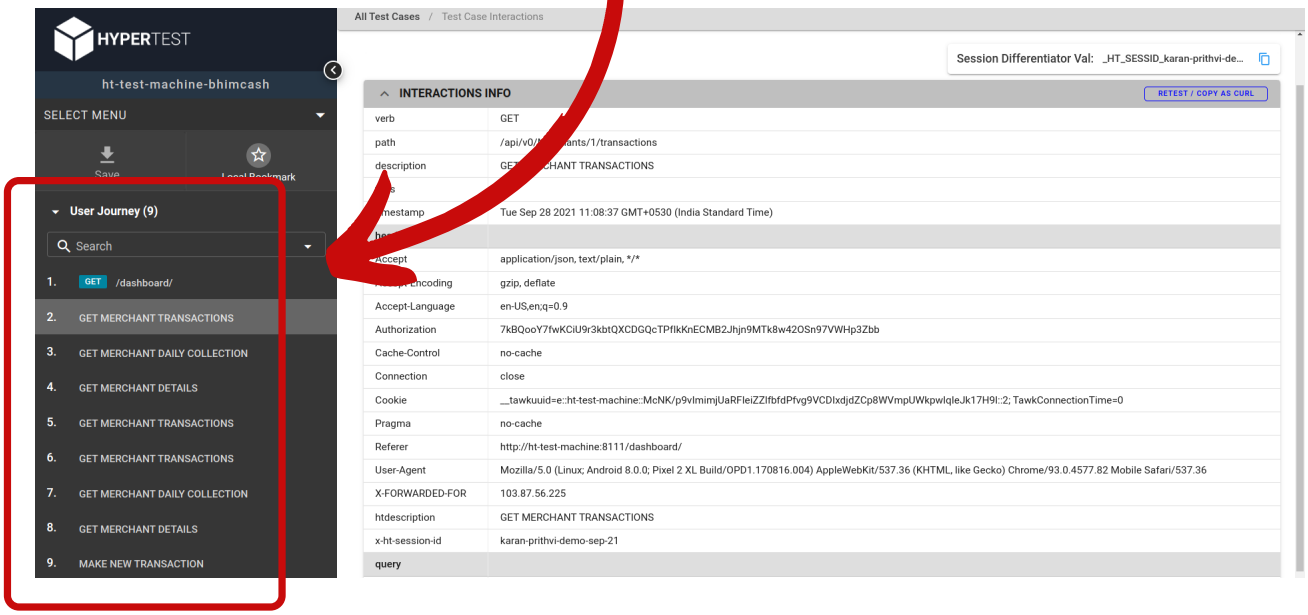
SUMMARY

In the modern API-first world, where APIs carry > 80% of all functional and business logic and change very rapidly, having an API automation suite with integration level tests has become a necessity. Such tests have the power to regress the services layer most extensively, catching and fixing the most critical errors pre-release.

HyperTest is designed to be the first autonomous API test platform of its kind that needs little to zero manual intervention from devs or QA in maintenance, building itself organically from network traffic and catching errors that no manually written test suite can. **Schedule a demo** to understand how HyperTest can get you ready with an API automation suite in days vs weeks.

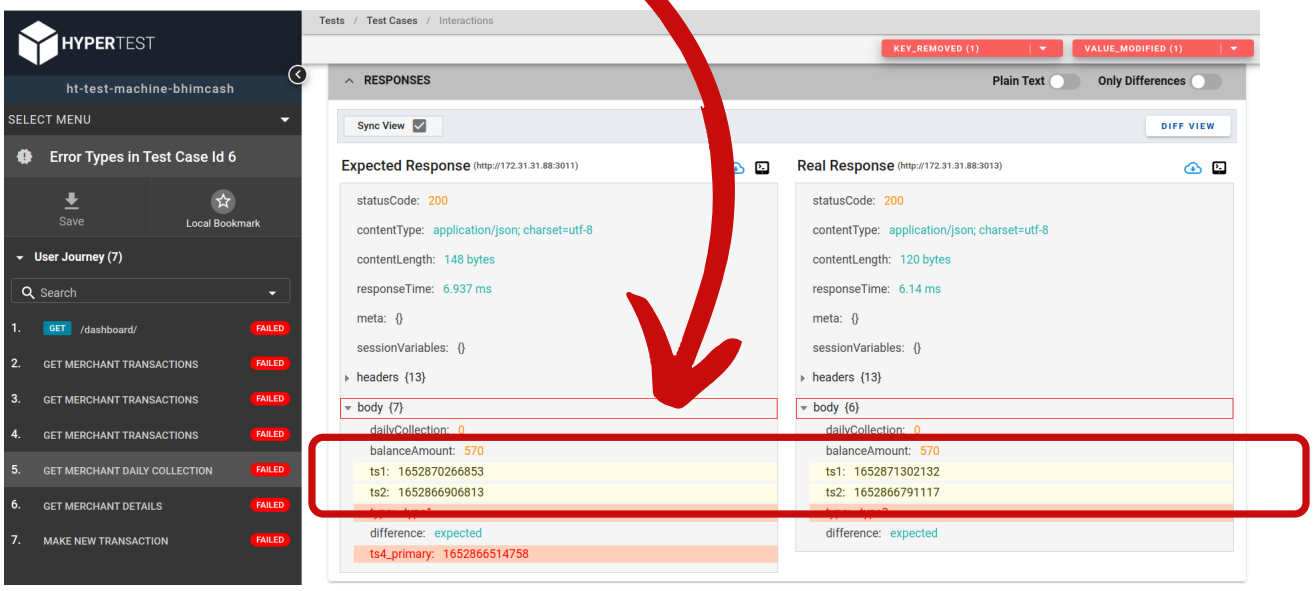
ANNEXURE A

The example below shows a sample test case. Starting on the left-hand side is the exact sequence of steps a typical user would take that is generated by the network traffic by HyperTest



ANNEXURE B

HyperTest runs a typical request several times on the 2 versions to build confidence about the response. Here in this case results marked in yellow are responses that differ between the two versions but is not a breaking change hence noise and marked in yellow.



ANNEXURE C

HyperTest creates and maintains an updated inventory of APIs. It makes a catalog of all API schemas with parameters, verbs, payloads and headers discovered via the network. Updates existing OAS files or generates new.