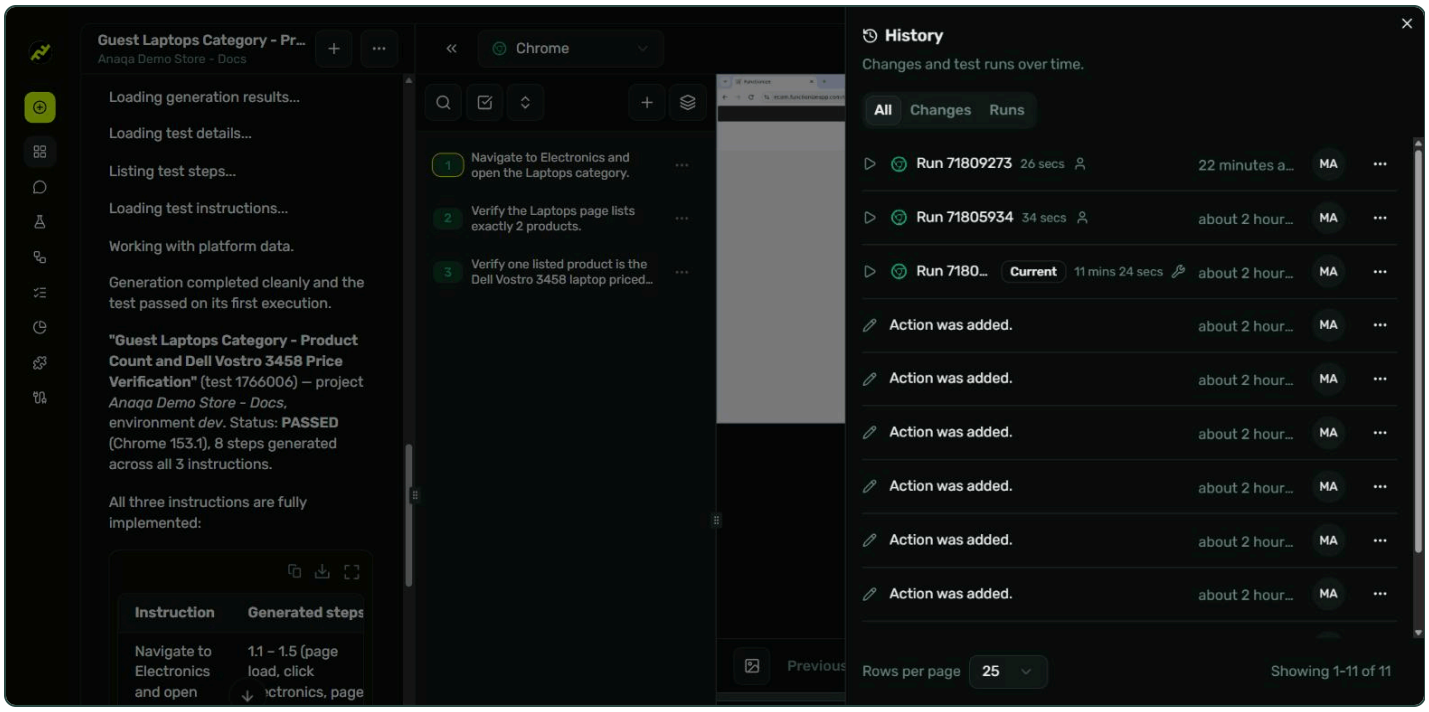
History: Every Change and Every Run, Recorded

Studio Basics · https://docs.functionize.com/hc/en-us/articles/43769790745495-History-Every-Change-and-Every-Run-Recorded

Every test in Studio keeps a record of what happened to it — every run, and every change to its steps, with who did it and when. It is behind one button in the test editor and it answers the question people usually cannot answer about automated tests: *why is this different from last week?*

Opening it

In the test editor, click the **history** icon in the top toolbar — the clock, sitting between the browser selector and Test Settings.



The panel is headed "*Changes and test runs over time*" and has three tabs:

All	Runs and changes together, newest first. The default, and usually what you want — it shows cause next to effect.
Changes	Only edits to the test.
Runs	Only executions.

Reading a run entry

Each run shows its **run ID**, how long it took, how long ago it was, and who triggered it. The run currently loaded in the editor is marked **Current**.

Two things this is good for:

Comparing durations. A run that suddenly takes four times as long is telling you something about your application even if it still passed. That is invisible on a pass/fail badge and obvious here.

Getting the run ID. When you raise a ticket, the run ID and its timestamp let us find the exact execution instead of asking you to reproduce it.

Every run entry also carries a small menu at its right:

View execution	Loads that run into the editor, so the steps, screenshots and Data tab all describe it. This is how you look at what happened three runs ago without leaving the test.
Latest execution	Jumps back to the most recent run. Useful once you have wandered several runs into the past and want to get back to now.
Delete execution	Removes that run from the record. Worth being reluctant about: the value of this timeline comes from it being complete, and a deleted run is a gap in the evidence somebody may want later.

Reading a change entry

Changes are recorded in plain language — "*Action was added.*", "*Test created.*" — each with a timestamp and the person responsible.

A change entry's menu holds one item, and it is the most important control in the panel: **Restore version**.

> **Restore version is the undo you did not know you had**
> It puts the test back to how it was at that point in the timeline. If an edit made a test worse — yours or the agent's — you do not have to reconstruct the earlier version from memory. Find the change that preceded the problem and restore what came before it.

Two things to know before you use it. **Restoring is itself a change**, so it lands on the timeline like any other and can be restored past in turn — you cannot paint yourself into a corner. And **a restored test has not been verified**: it is in a state that once worked, against an application that has moved since. Run it.

This is the audit trail. It answers "who changed this and when", which on a conventional automation suite usually means digging through version control, if the change was committed at all. Here it includes changes the agent made on your behalf during a repair.

Why this matters more than it sounds

The common failure mode with automated tests is not a test that breaks. It is a test that quietly becomes wrong — an assertion loosened to make a failure go away, a step deleted to get a build green — and nobody remembers doing it.

Because Studio records both the changes and the runs on one timeline, you can see a test's behavior change and the edit that preceded it, next to each other. If a test started passing suspiciously soon after an edit, the history shows you that in one glance.

This pairs directly with the distinction in Diagnosing a Failed Test: a repair that fixed genuine drift is healthy, and an expectation quietly rewritten to match a broken page is not. History is where you tell them apart after the fact.

What to do with it

1. **Before trusting a long-green test**, glance at Changes. A test that has been edited repeatedly and passes every time deserves a look.
2. **When a test starts failing**, check whether a change preceded it. Often the application did not break — the test moved.
3. **When durations drift upward**, treat it as a finding about the application, not about the test.
4. **When raising a ticket**, copy the run ID.

The conversation that produced each change is kept too, in the agent chat beside the steps. The history tells you *what* changed; the session tells you *why*. See Exercise 4 — Change a Test by Asking. Every session a test has ever carried is listed on the test itself — see Sessions: The Conversations Behind Your Tests.