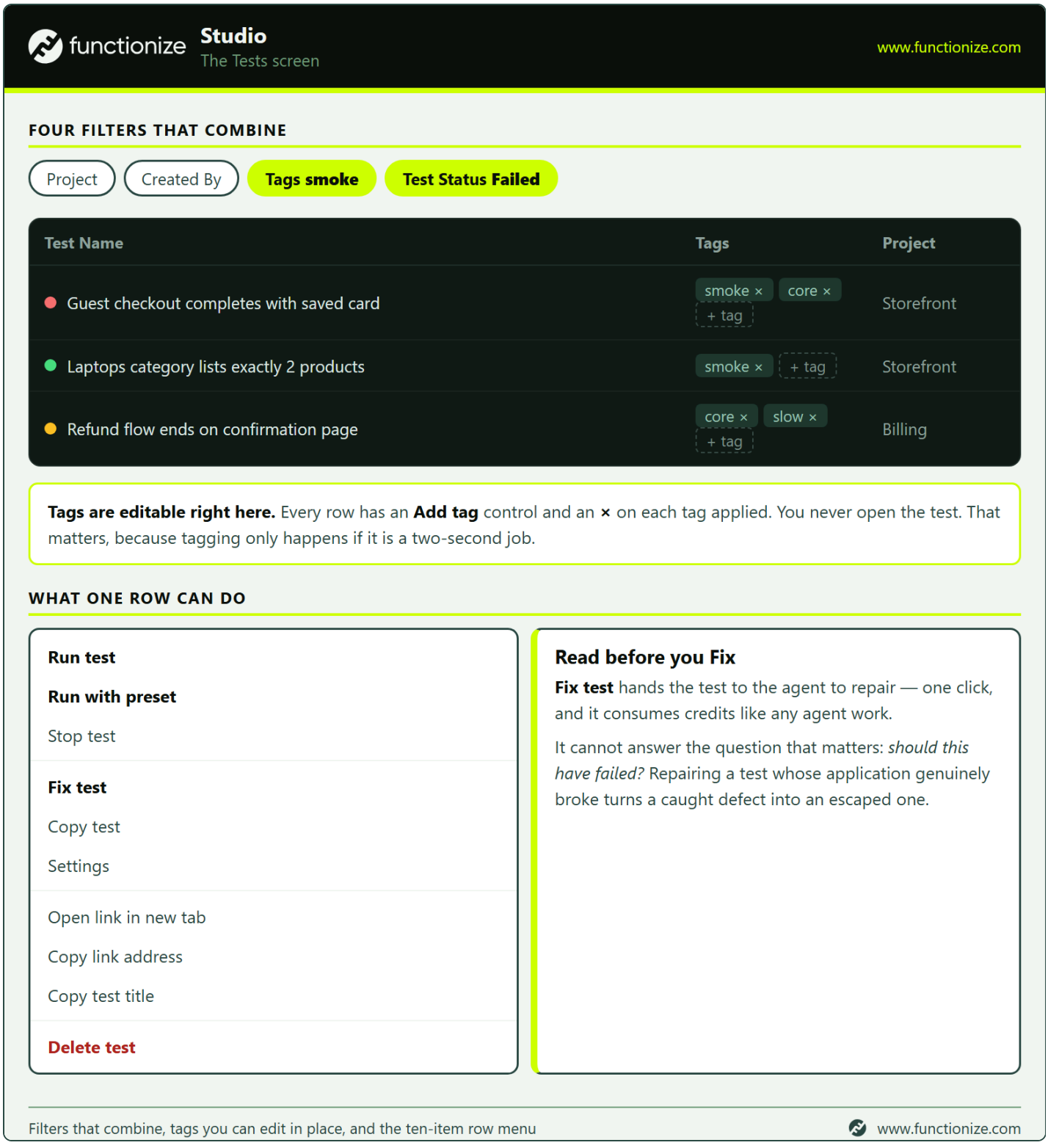
The Tests Screen: Finding, Tagging and Fixing Tests at Scale

Features · https://docs.functionize.com/hc/en-us/articles/43802539697431-The-Tests-Screen-Finding-Tagging-and-Fixing-Tests-at-Scale

Every test you or anyone else has built lives on one screen. For the first couple of weeks that screen is a convenience. By the time an account holds a few thousand tests it is the main way anybody finds anything, and the habits you form early decide whether it stays usable.

This page covers what is on it, the four filters, the thing most people miss — tags are editable right here — and the ten-item menu on every row.



What a row tells you

Open **Tests** in the left sidebar. Each row carries six pieces of information, and all of them are useful for different questions.

Status	The result of the last run, shown as a word before the name: **Passed**, **Failed**, **Warning** or **Running**.

Test Name	What it proves, if whoever built it named it well. See *Naming, Tagging and Not Drowning*.
Tags	Editable in place. This is the important one — see below.
Project	Which project it belongs to. A test lives in exactly one.
Updated / **Last Run**	Two different questions. A test updated last week but not run since last month is a test somebody edited and never verified.
Browsers	Browser icons, one per browser the test has results for. A test showing a single icon is not cross-browser coverage, whatever the suite is called.

The four filters

These sit above the table and combine, which is what makes them worth learning.

Project	Narrow to one application. Usually the first filter you reach for.
Created By	Whose tests these are. Useful when somebody leaves and their work needs an owner.
Tags	A searchable list of every tag in the account. This is how you assemble a view of a suite that spans projects.
Test Status	**Passed**, **Failed**, **Warning** or **Running**.

Three filter combinations worth memorizing
Status = Failed on its own is your morning triage list.
Tags = smoke plus **Status = Failed** tells you whether anything that gates a release is broken.
Created By = someone who has left, sorted by Last Run, finds the tests nobody is looking after.

Tags are editable here, and tags are account-wide

Each row's Tags cell has a small **Add tag** control and an **×** on every tag already applied. You do not have to open a test, find its settings and save to change a tag. That matters because tagging is the single highest-leverage habit in Studio and anything that makes it slower means it does not happen.

Two properties are worth being precise about:

- **Tags are account-wide, not per-project.** The Tags filter offers every tag anyone has ever created, across every project. A tag called `smoke` means whatever your whole organization has collectively decided it means.
- **Tags are how a suite stays current.** A hand-picked orchestration is right the day you build it and quietly wrong a month later. A tag scheme lets you assemble by what tests *are* rather than by remembering which ones to tick.

Because tags are shared, agreeing a small vocabulary before a second team starts using them is worth ten minutes. Three or four tags that everyone uses beat forty that nobody does.

The row menu

The **…** at the end of a row holds ten items. Four of them do work you would otherwise go somewhere else for.

Run test	Starts it now, without opening the editor.
Run with preset	Runs it against a chosen execution preset — "*Choose which execution preset to apply for this run.*" This is how you check one test against a different data set without changing anything. See Execution Presets: Running One Suite Against Different Data.
Stop test	Halts a run in flight.
Fix test	Hands the test to the agent to repair. It is the same repair the agent does inside a session, started from the list — so it is agent work and it consumes credits like any other agent work.
Copy test	Duplicates it. The fastest way to build a variant.
Settings	Jumps straight to that test's settings. See Settings: Where Everything Lives and What to Change First.
Open in new tab / **Copy link address** / **Copy test title**	A test has a stable URL. Paste it into the ticket rather than describing which test you mean.
Delete test	Removes it. Any orchestration that included it simply has one fewer test.

A word about Fix test

One click that repairs a broken test is genuinely useful and genuinely easy to misuse. The question it cannot answer for you is the one that matters: *should this test have failed?*

If the application changed legitimately and the test is now looking for something that moved, repairing it is exactly right. If the application broke, repairing the test makes the failure disappear without fixing anything — and you have converted a caught defect into an escaped one.

Read the failure before you press Fix. Diagnosing a Failed Test in Functionize Studio is the ten minutes that makes the difference, and *Reviewing AI-Generated Tests* covers what to check on the repair afterward.

Creating a test from here

New Test at the top right starts the same agent conversation the Home screen does, already pointed at a project. If you know which application you are testing, starting here saves a step. See Starting a Session: The Studio Home Screen.

Three habits

- **Tag as you build, not later.** Retro-tagging four hundred tests is a job nobody does. Tagging one test as you finish it costs nothing.
- **Sort by Last Run, ascending, once a month.** The bottom of that list is tests nobody runs. Either they matter and should be in an orchestration, or they do not and should go.
- **Filter by Created By when someone changes team.** Ownership is the thing that silently lapses, and this screen is the only place it is visible.

Where to go next

- Naming, Tagging and Not Drowning: Conventions for a Shared Account — what to agree before the list gets long.
- Orchestrations: Running a Suite as One Job — turning a tagged set of tests into something that runs.
- Diagnosing a Failed Test in Functionize Studio — what to do before you press Fix.
- Projects: Boundaries, Defaults and What Inherits What — why a test belongs to exactly one project.