Your First Test in Functionize Studio: A Visual Walkthrough

Studio Basics · https://docs.functionize.com/hc/en-us/articles/43766886706839-Your-First-Test-in-Functionize-Studio-A-Visual-Walkthrough

This is a picture-by-picture walkthrough of creating your first test in Functionize Studio, from an empty project to a green run. Every screen below is the real product, working against the Functionize demo store, so you can follow along exactly without touching your own application. Allow about ten minutes.

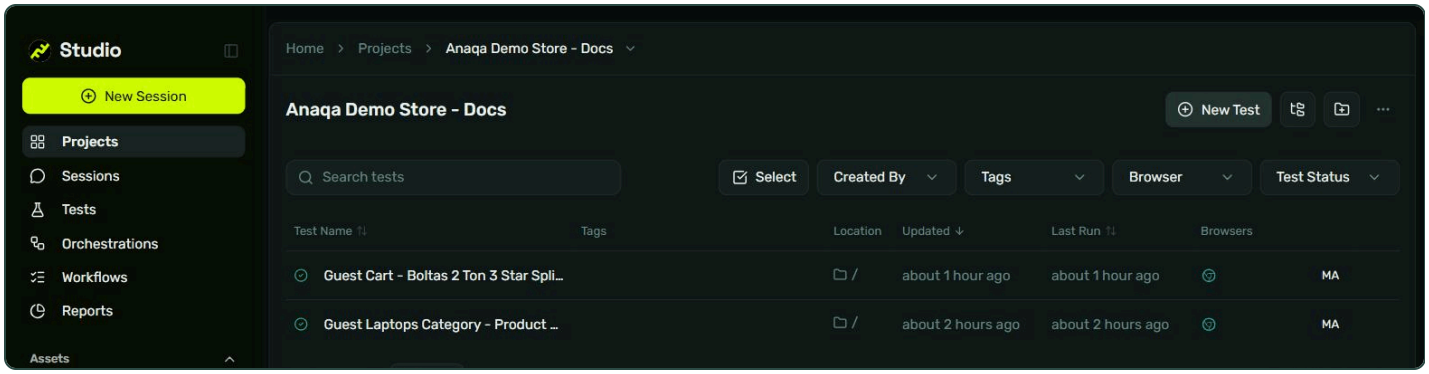
What you need

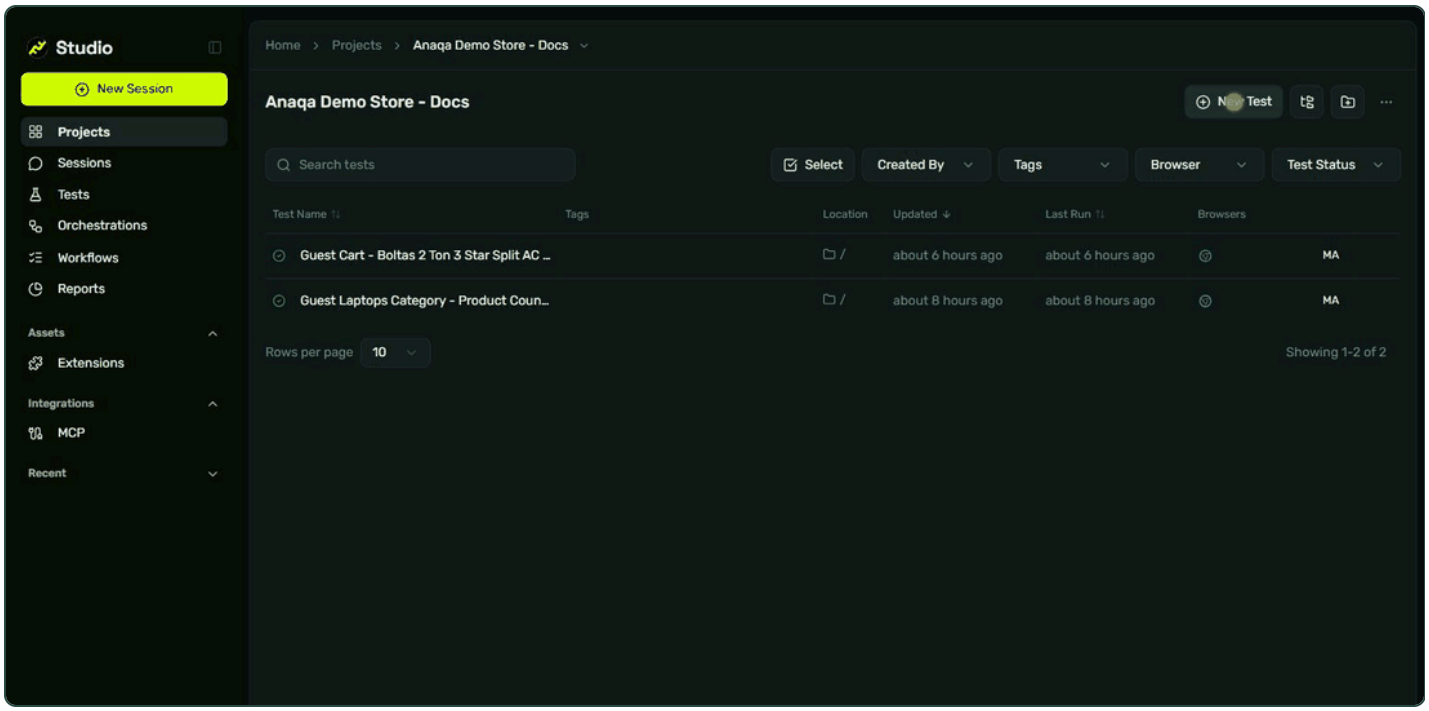
- A Functionize Studio account, signed in at `studio.functionize.com`.
- The Anaqa demo store, our own practice site: `https://ecom.functionizeapp.com/testsites/anaqa/index.php?route=common/home`

Nothing to install, and **nothing to sign in to** — everything in this walkthrough uses the store as a guest, so there are no credentials to manage.

Step 1 — Make a project

Projects group related tests. From **Projects** in the left sidebar choose **New Project**, name it, and open it. Here is one with two tests already in it:

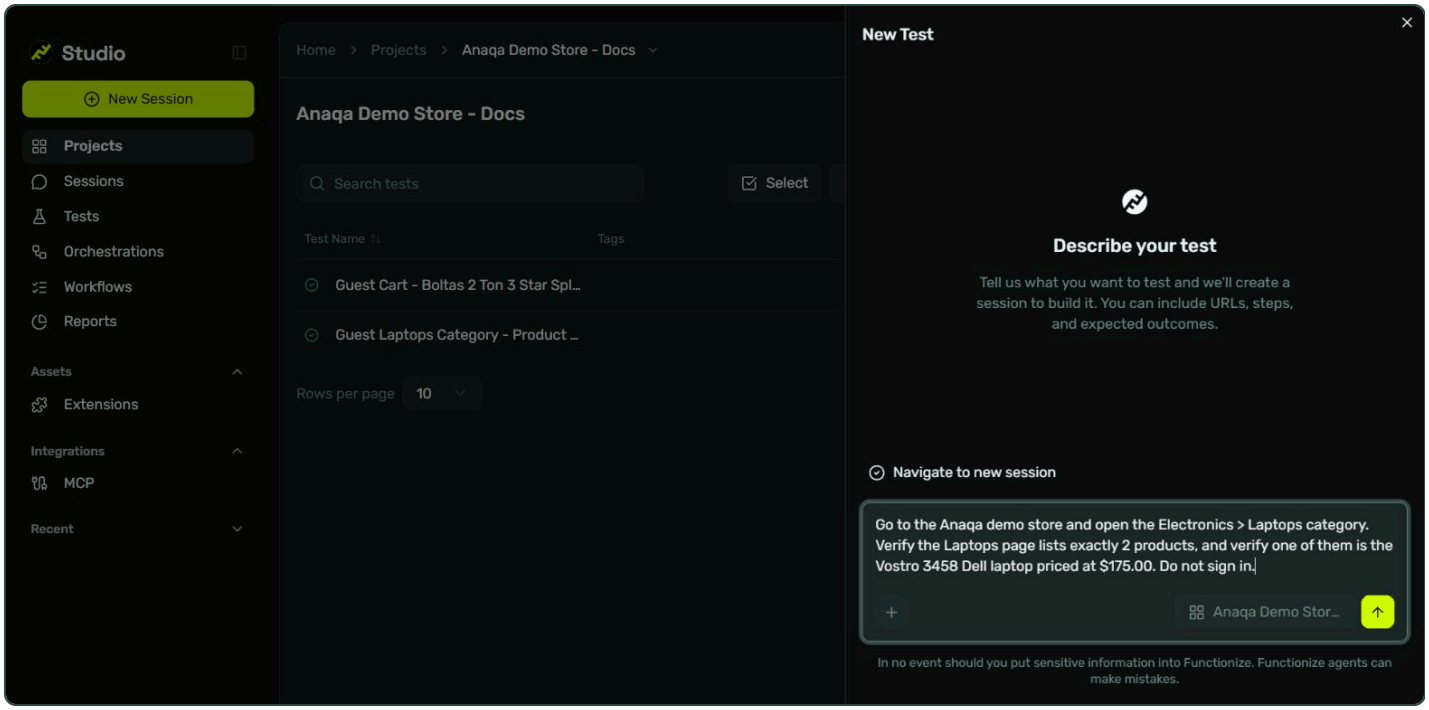




Opening New Test and describing what you want.

Step 2 — Describe the test

Click **New test**. A panel opens asking what you want. You are not writing steps here — you are describing an outcome, the way you would brief a colleague.



The description used was:

> Go to the Anaqa demo store and open the Electronics > Laptops category. Verify the Laptops page lists exactly 2 products, and verify one of them is the Vostro 3458 Dell laptop priced at $175.00. Do not sign in.

Three things make a description work:

- **Give a starting URL.** The agent checks it is reachable before building anything.
- **Say what a user does, not how the page is built.** "Open the Laptops category" beats naming a CSS class. Studio finds elements visually and contextually, so describing intent is what lets a test survive a redesign.
- **Say what "passed" means.** "Lists exactly 2 products" is a real assertion. Without one you get a test that clicks around and proves nothing.

Check the project chip at the bottom of the box before you send — it confirms where the test will be created.

Step 3 — Approve the plan

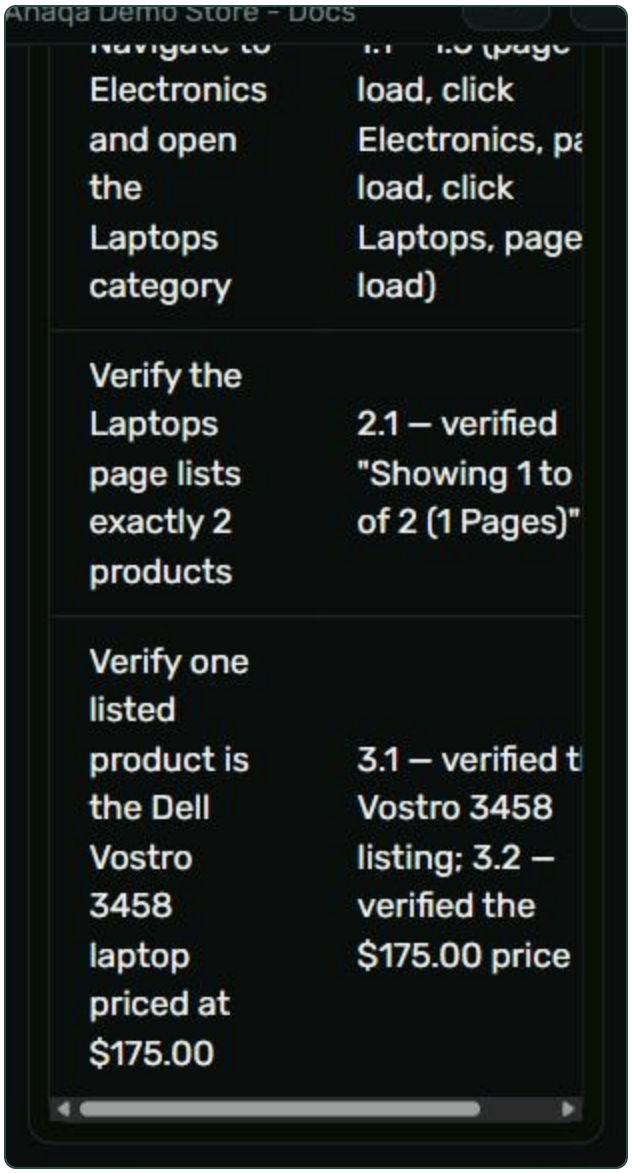
Studio does not start building straight away. It turns your paragraph into numbered instructions and shows them to you first. Read them: this is the cheapest moment to catch a misunderstanding.

If something is wrong, say so in plain language and it will redraft. When it looks right, reply **Yes, create it**.

Step 4 — Watch it build

Studio starts a background agent and narrates as it goes — checking the URL, resolving the environment, then generating and verifying each step against the live page. Generation usually takes between thirty seconds and a couple of minutes.

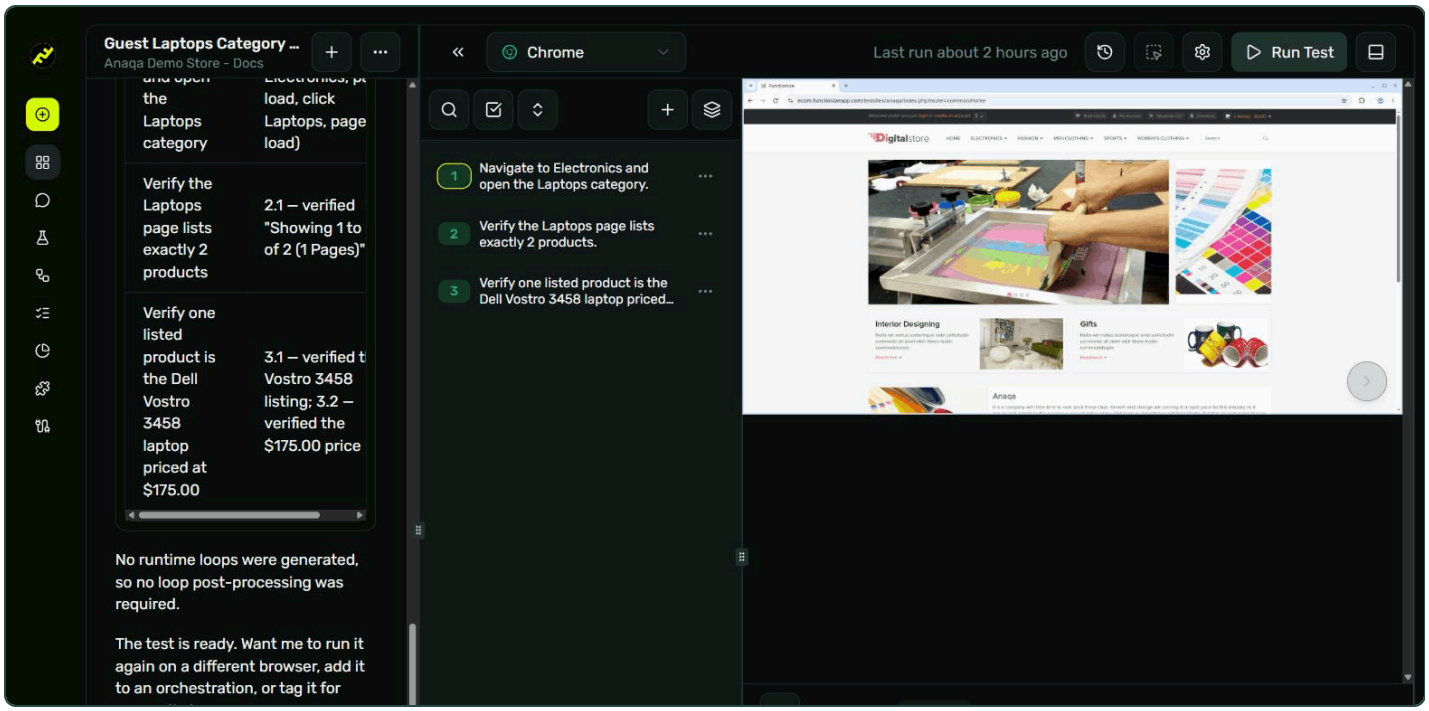
When it finishes it reports what it made:



On this run, **five instructions became ten steps, and every step passed**. That expansion is normal — "open the product" alone involves loading a category page, finding the item and clicking through to it.

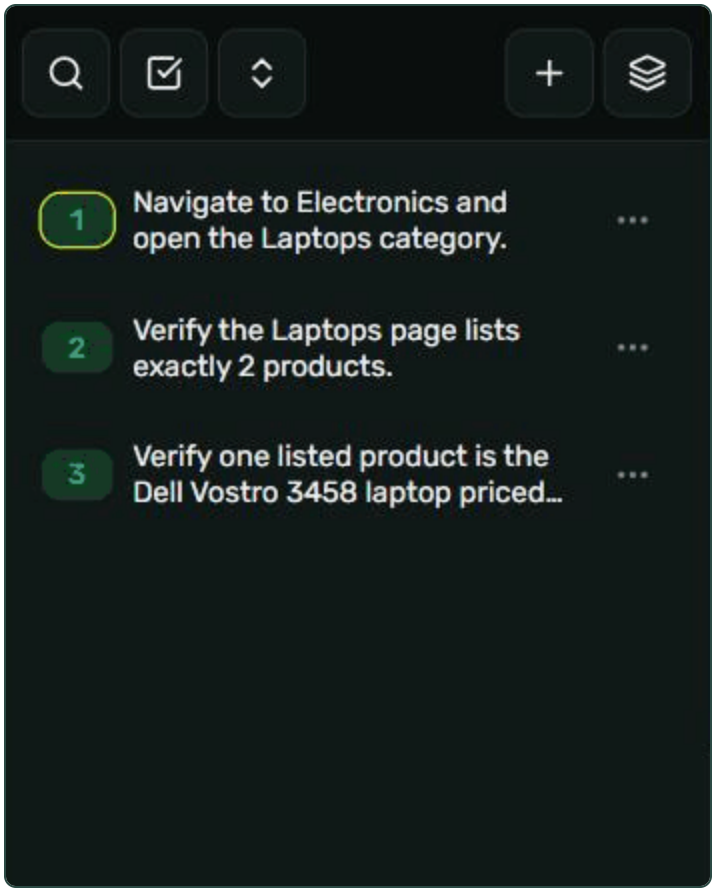
Step 5 — Read the test editor

The editor is three panes, and knowing what each is for makes the rest of the product obvious:



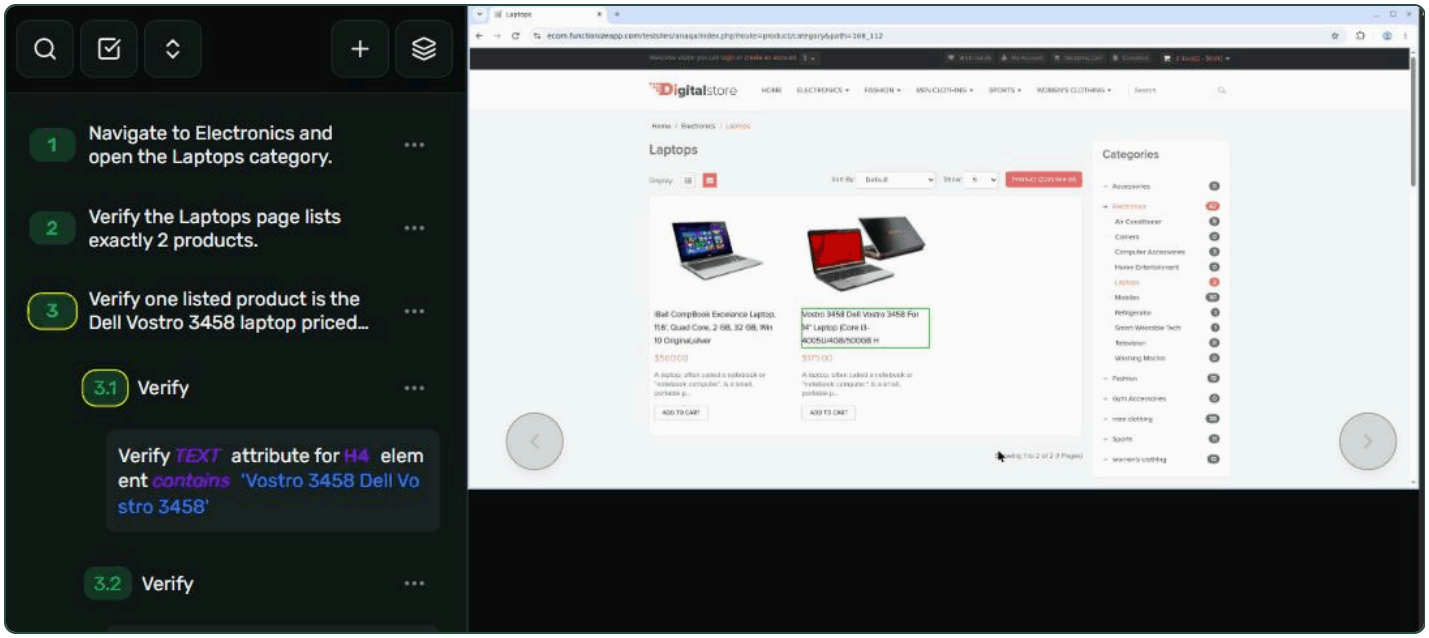
- **Left — the agent.** The conversation that built this test, and where you ask for changes. It stays with the test, so the reasoning behind a step is still there weeks later.
- **Middle — the steps.** Your instructions, numbered and color-coded by result.
- **Right — the browser and the data.** The page at the selected step, and the values that step read.

The steps pane on its own:



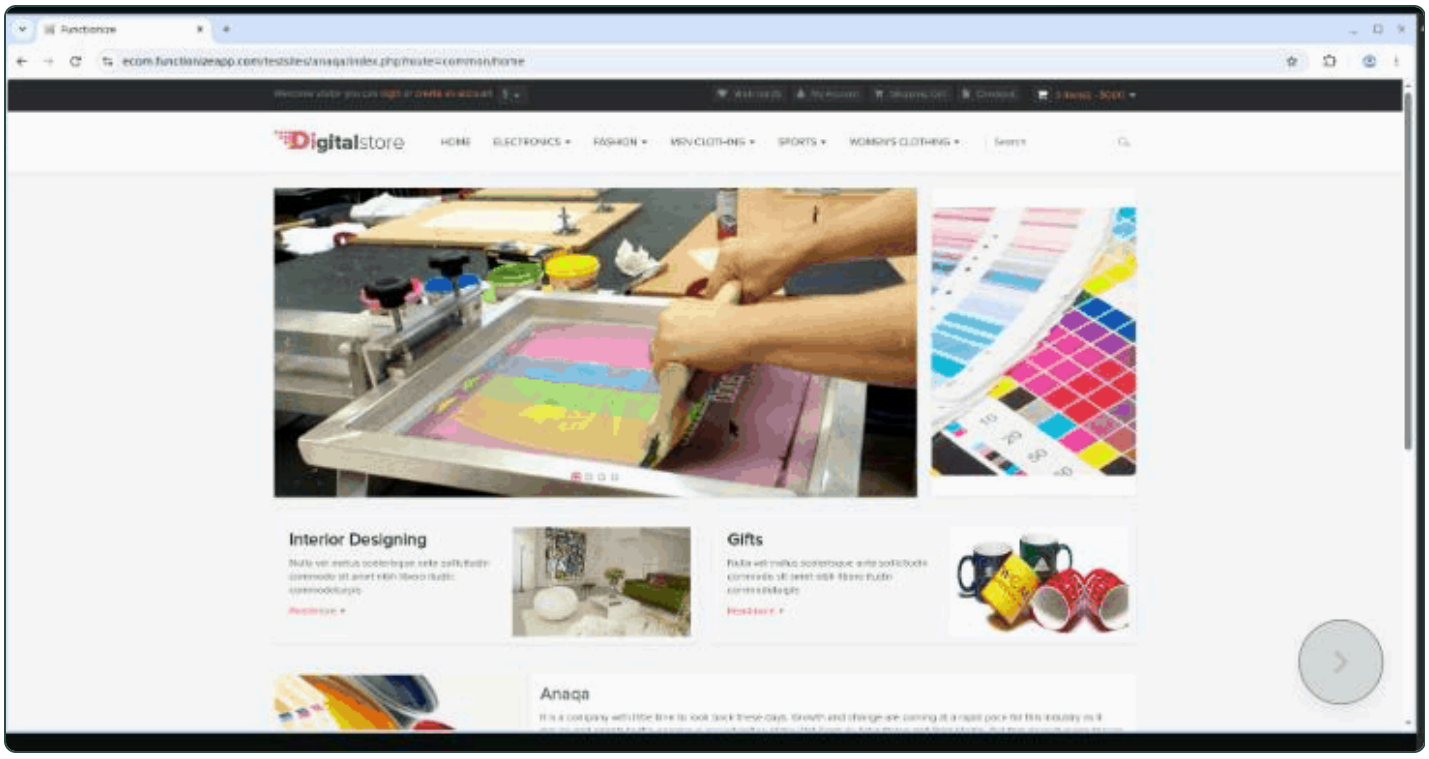
Step 6 — Open a step

Click any step to expand it and see the action underneath:



The action reads `Verify TEXT attribute for SPAN element contains '$966.00'` — a description of the target, not a brittle selector path. That is what keeps the step working when the page around it changes.

The right-hand pane shows the store as it was during the run:



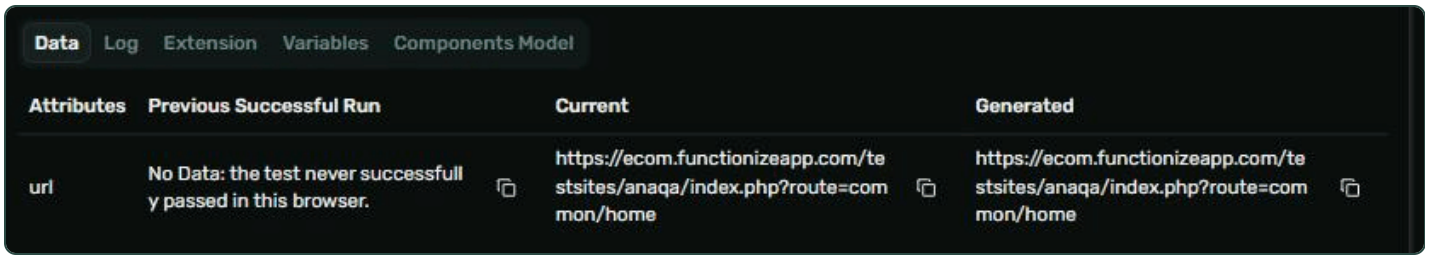
Step 7 — Run it

Choose a browser from the selector at the top, then click **Run Test**. Studio boots a clean runtime host and executes the test there — not in your browser, so you can close the tab.

Because each run starts clean, there is no leftover cart or cookie from last time unless the test creates one itself.

Where to look when something changes

Under the browser pane, the **Data** tab compares three columns:



- **Previous Successful Run** — the value when the test last passed.
- **Current** — the value in this run.
- **Generated** — the value when the test was created.

That comparison answers most failures on its own. Learn it now, while everything is green.

What to do next

- Work through the **Exercises** section — seven short, self-paced exercises on this same demo store.
- Group tests into an **Orchestration** to run them together on a schedule.
- Connect your own AI tool over MCP so you can create and check tests from your editor.
- Ask us to build a **skill** for your application, so the agent starts out knowing your domain rather than discovering it.

If something here does not match what you see, the product has moved on and we would like to know — raise a ticket and we will correct the article.

Where to go next

- Understanding a Test Run and Its Results in Functionize Studio — read what it proved
- Exercise 1 - Create Your First Test From a Prompt — do it yourself, on our demo store
- Writing Prompts That Produce Durable Tests — prompts that still work next quarter