

Test your store from Claude

Anthropic's webapp-testing skill for exploring the purchase path with Playwright, and how to turn that exploration into a test that runs in your CI. With a demo store with no payments and its test, run locally.

FOR	Claude Code and projects on Node 18 or newer
TESTED WITH	@playwright/test 1.62 · Node 22 · Windows 11 · the skill was not run
UPDATED	26 October 2026
DOCUMENTATION	github.com/anthropics/skills · playwright.dev
ARTICLE	ecomlabs.dev/blog/claude-webapp-testing-playwright
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before
you start

What you will set up

Step 01

Start the store and its test

Step 02

Install the skill

Step 03

Explore the path

Step 04

Turn it into a test

Step 05

Check the test catches failures

Step 06

Take it to your CI

Limits

Data, payments and production

If
something
fails

Troubleshooting

When you
finish

Checklist

What you will set up

The webapp-testing skill gives Claude Code a method for testing local web apps: it writes Playwright scripts in Python, starts the server with `scripts/with_server.py`, waits for `networkidle`, takes screenshots and picks selectors before acting. This guide uses it to explore the product, cart and checkout path, and takes that path into a `@playwright/test` test that runs in CI.

CAUTION

The store's test was run locally. We did not run the skill: it needs Playwright for Python, which we did not install on the test machine. The workflow is validated, but we did not run it on GitHub.

What is in the ZIP

FILE	WHAT IT DOES
<code>server.js</code>	Serves the store on port 4173, no dependencies.
<code>public/</code>	Home, product, cart and checkout with no payments.
<code>tests/purchase.spec.js</code>	The purchase path with selectors by role and text.
<code>playwright.config.js</code>	Starts the store before the tests.
<code>.github/workflows/e2e.yml</code>	Runs the tests on every PR.

Requirements

- Node 18 or newer (tested with Node 22).
- For the skill: Python and Playwright for Python.

Start the store and its test

```
npm install
npx playwright install chromium
npm test
```

Both tests should pass. `npm start` serves the store at `http://localhost:4173` so you can see it in the browser. The server:

`server.js`

```
// Minimal server for the demo store: serves public/ with Node, no dependencies.
import { createServer } from "node:http";
import { readFile } from "node:fs/promises";
import { extname, join, normalize, sep } from "node:path";
```

```
const root = join(import.meta.dirname, "public");
const types = { ".html": "text/html; charset=utf-8", ".js": "text/javascript; charset=utf-8" };
const port = Number(process.env.PORT ?? 4173);

createServer(async (req, res) => {
  const path = new URL(req.url, "http://localhost").pathname;
  const file = normalize(join(root, path === "/" ? "index.html" : path));
  if (!file.startsWith(root + sep)) {
    res.writeHead(403).end();
    return;
  }
  try {
    const body = await readFile(file);
    res.writeHead(200, { "content-type": types[extname(file)] ?? "application/octet-stream" }).end(body);
  } catch {
    res.writeHead(404, { "content-type": "text/plain; charset=utf-8" }).end("Not found");
  }
}).listen(port, () => console.log(`Demo store at http://localhost:${port}`));
```

It checks that the requested path stays inside `public/` before reading the file, so a URL with `../` serves nothing from outside.

STEP 02

Install the skill

```
pip install playwright
playwright install chromium
```

And inside Claude Code:

```
/plugin marketplace add anthropics/skills
/plugin install example-skills@anthropic-agent-skills
```

The `example-skills` plugin brings twelve skills, webapp-testing among them. The repository warns they are examples for demonstration and educational purposes: read the `SKILL.md` before using it.

STEP 03

Explore the path

Ask for the exploration with the store stopped; per its `SKILL.md`, the skill starts it with `with_server.py`:

```
Start the store with "node server.js" on port 4173. Walk through it: open the Basic T-shirt,
add it to the cart, go to checkout and confirm with the email test@example.com. Take a
screenshot at each step and tell me which selectors you would use for a test.
```

Check the screenshots and the selector list. Ask for roles and visible text ("the Add to cart button") and drop CSS classes or positions. The exploration scripts are not kept: they are for looking.

STEP 04

Turn it into a test

Turn that path into a `@playwright/test` test in `tests/`, with selectors by role and text.

The demo store's test looks like this:

`tests/purchase.spec.js`

```
// Purchase path in the demo store: product, cart and checkout.
// It uses roles and visible text, like the ones exploration finds.
import { expect, test } from "@playwright/test";

test("product, cart and test checkout", async ({ page }) => {
  await page.goto("/");
  await page.getByRole("link", { name: "Basic T-shirt" }).click();

  await page.getByRole("button", { name: "Add to cart" }).click();
  await expect(page.getByRole("status")).toHaveText("Added to cart");
  await expect(page.getByRole("link", { name: "Cart (1)" })).toBeVisible();

  await page.getByRole("link", { name: "Cart (1)" }).click();
  await expect(page.getByRole("cell", { name: "Basic T-shirt" })).toBeVisible();
  await expect(page.getByRole("cell", { name: "€19.99" })).toBeVisible();

  await page.getByRole("link", { name: "Go to checkout" }).click();
  await page.getByLabel("Email").fill("test@example.com");
  await page.getByRole("button", { name: "Place test order" }).click();

  await expect(page.getByRole("heading", { name: "Test order confirmed" })).toBeVisible();
  await expect(page.getByRole("link", { name: "Cart (0)" })).toBeVisible();
});

test("an empty cart does not lead to checkout", async ({ page }) => {
  await page.goto("/cart.html");
  await expect(page.getByText("Your cart is empty.")).toBeVisible();
  await expect(page.getByRole("link", { name: "Go to checkout" })).toBeHidden();
});
```

And the config, which starts the store before the tests:

`playwright.config.js`

```
import { defineConfig, devices } from "@playwright/test";

export default defineConfig({
  testDir: "tests",
```

```

retries: process.env.CI ? 1 : 0,
use: { baseUrl: "http://localhost:4173", trace: "on-first-retry" },
// Playwright starts the store before the tests and stops it afterwards.
webServer: { command: "node server.js", url: "http://localhost:4173", reuseExistingServer: !process.env.CI
},
projects: [{ name: "chromium", use: { ...devices["Desktop Chrome"] } }],
});

```

STEP 05

Check the test catches failures

A test that never fails proves nothing. In `public/store.js`, delete the `save([]);` line near the end, which empties the cart after the order:

`public/store.js`

```

// The cart lives in localStorage: enough for a demo store with no backend.
const read = () => JSON.parse(localStorage.getItem("cart") ?? "[]");
const save = (cart) => localStorage.setItem("cart", JSON.stringify(cart));
const euros = (cents) => "€" + (cents / 100).toFixed(2);

const counter = document.getElementById("cart");
const paintCounter = () => (counter.textContent = `Cart (${read().length})`);
paintCounter();

document.getElementById("add")?.addEventListener("click", () => {
  save([...read(), { name: "Basic T-shirt", price: 1999 }]);
  paintCounter();
  document.getElementById("notice").textContent = "Added to cart";
});

const rows = document.getElementById("rows");
if (rows) {
  const cart = read();
  for (const line of cart) {
    const row = rows.insertRow();
    row.insertCell().textContent = line.name;
    row.insertCell().textContent = euros(line.price);
  }
  document.getElementById("empty").hidden = cart.length > 0;
  document.getElementById("to-checkout").hidden = cart.length === 0;
}

document.getElementById("checkout")?.addEventListener("submit", (event) => {
  event.preventDefault();
  save([]);
  paintCounter();
  document.querySelector("main").innerHTML =

```

```
"<h1>Test order confirmed</h1><p>Nothing was charged: this store has no payment gateway.</p>";
});
```

Run `npm test`. In our run (on the Spanish store), the purchase test failed like this:

```
Error: expect(locator).toBeVisible() failed
Locator: getByRole('link', { name: 'Carrito (0)' })
Expected: visible
Error: element(s) not found
```

Put the line back and run it again: both tests should pass.

STEP 06

Take it to your CI

`.github/workflows/e2e.yml`

```
# Runs the purchase path with Playwright on every PR and every push to main.
name: E2E

on:
  pull_request:
  push:
    branches: [main]

jobs:
  e2e:
    runs-on: ubuntu-latest
    timeout-minutes: 15
    permissions:
      contents: read
    steps:
      - uses: actions/checkout@v6
      - uses: actions/setup-node@v7
        with:
          node-version: 22
      - run: npm ci
      - run: npx playwright install --with-deps chromium
      - run: npx playwright test
```

`npm ci` needs the `package-lock.json` that `npm install` creates: commit it. In CI, `reuseExistingServer` is false and Playwright always starts its own store.

LIMITS

Data, payments and production

- **Made-up data:** `test@example.com`, never customer data.

- **No real payments:** your gateway's test mode; on Shopify, a development store with test orders.
- **Never against production:** it creates orders, takes stock, sends emails and pollutes analytics.
- **Password:** Shopify development stores usually have one; keep it in a CI secret.

IF SOMETHING FAILS

Troubleshooting

SYMPTOM	LIKELY CAUSE	FIX
"Executable doesn't exist"	The browser is missing	<code>npx playwright install chromium</code>
The test fails on the connection	The store did not start	Check <code>webServer</code> and port 4173
The skill is not used	Plugin not installed	Install <code>example-skills</code> and restart
The skill's scripts fail	Playwright for Python is missing	<code>pip install playwright</code>
<code>npm ci</code> fails in CI	No <code>package-lock.json</code>	Run <code>npm install</code> and commit it

WHEN YOU FINISH

Checklist

- ☐ `npm test` passes with the store intact.
- ☐ The test fails if you remove the cart clearing.
- ☐ Selectors are by role and visible text.
- ☐ Data is made up and there are no real payments.
- ☐ `e2e.yml` runs on every PR.