Green isn't proof.

Verifying React E2E tests written by AI.

A Promise is truthy. A button may be enabled.

Ghost PR #28712, merged.

Before

```
await expect(likeButton).toHaveText('1');
expect(likeButton.isDisabled())
  .toBeTruthy();
```

After

```
await expect(likeButton).toHaveText('1');
await expect(likeButton).toBeDisabled();
```

Assert the state, not the Promise.

Element gone. Does this fail?

Cal.com PR #28486, merged.

Before, line 203

```
await expect(
  page.getByTestId("away-emoji"))
  .toBeTruthy();
```

After

```
await expect(
  page.getByTestId("away-emoji")
  .first()).toBeVisible();
```

Playwright 1.62.1

```
✓  1 tests/locator-quiz.spec.mjs:8:1 › toBeTruthy on a Locator (64ms)
✘  2 tests/locator-quiz.spec.mjs:14:1 › toBeVisible on the element (2.1s)
Error: element(s) not found
```

toBeTruthy checks the Locator object.

Suppose a refactor drops the POST.

src/LikeButton.jsx

```
  setLiked(true);
  setSaving(true);
  setError("");
  try {
    await postLike();
  } catch {
```

Play the demo (22 s) on Google Drive

The screen passes. The missing request is caught.

Screen test, PASS

```
await like.click();
await expect(like)
  .toHaveAttribute(
    "aria-pressed", "true");
await expect(page.locator(".likes"))
  .toHaveText("129 likes");
await expect(page.getByRole("status"))
  .toHaveText("Liked!");
```

Request test, FAIL

```
const post = page.waitForRequest(
  (request) =>
    request.method() === "POST" &&
    new URL(request.url()).pathname
    === "/api/like",
  { timeout: 3000 },
);
await like.click();
await post;
```

1. Required evidence
POST sent.

2. Behavior removed
POST removed.

3. Why it failed
waitForRequest timed out.

A Locator is not the result. Await the check.

MUI X #22982 and Bruno #8317, merged.

MUI X, before

```
expect(page.getByText(
  '1/31/2025, 4:05:00 PM'))
  .not.to.equal(null);
```

MUI X, after

```
await page.getByText(
  '1/31/2025, 4:05:00 PM')
  .waitFor();
```

Bruno, before

```
expect(
  page.locator('span.item-name')
  .filter({ hasText: BRU_REQ_NAME }))
  .toBeVisible();
```

Bruno, after

```
await expect(
  page.locator('span.item-name')
  .filter({ hasText: BRU_REQ_NAME }))
  .toBeVisible();
```

Wait for the result; await the assertion.

Did the same user remain inactive?

Cypress cy.prompt() documentation example.

The scenario

“User cannot be deactivated if already inactive.”

Example output

```
cy.get('#user-list div:nth-child(3) > .user-actions > .btn-danger').click();
cy.get('#user-list .status-inactive').should('contain', 'inactive');
```

Tie the assertion to the same user.

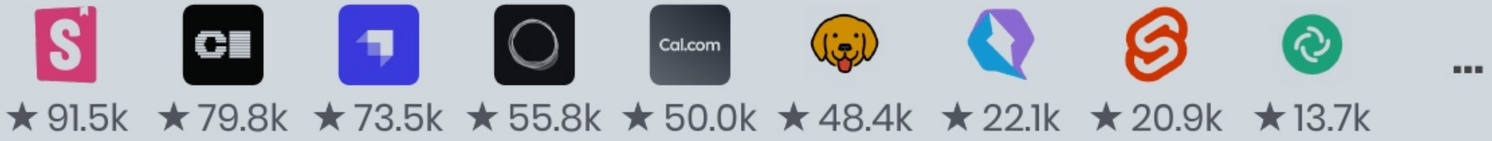
How common is this?

AI source review of 200 selected React repos. Not a defect rate.

34 of 200

Repos had a test that can pass while its feature is broken.

15 fixes merged upstream.



What e2e-skills does

Rules file	Scanner	Reviewer
Asks for failure paths.	Flags mechanical mistakes.	Asks what proves a save.

Rules come before writing. Checks come after.

The reviewer asks what proves the save.

Clean copy of my demo app, Claude. 3 runs each.

Plain review	Reviewer 3.5.3
0 of 3	**3 of 3**

Runs that flagged the missing POST.

Running the test without the POST is the proof.

Now inspect a test written by AI.

Which condition did it reach?

Screen

Request
demo

What did its assertion observe?

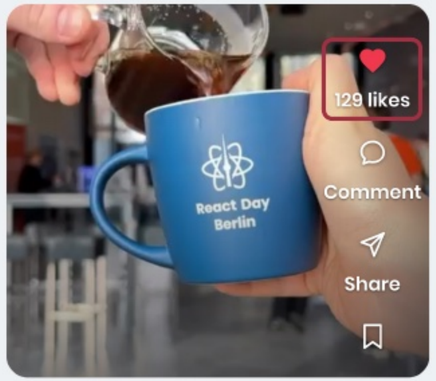
Server

Rollback
next

Remove recovery. Rejection still looks saved.

HTTP 500 on save.

Still pressed



Removed: undo and error

```
try {
  await postLike();
} catch {
  setLiked(previousLiked);
  setError("Couldn't save. Try again.");
} finally {
  setSaving(false);
}
```

The successful save path remains.

Does this success test protect recovery?

Written by Codex, no rules.

Generated success test

```
19 await like.click();
20 expect((await saved).status()).toBe(200);
21 await expect(like).toBeEnabled();
22 await expect(like).toHaveAttribute("aria-pressed", "true");
   …
27 await page.reload();
28 await expect(like).toBeEnabled();
29 await expect(like).toHaveAttribute("aria-pressed", "true");
```

Also promised: rejected save, then undo and error.

Recovery removed. Still PASS.

Written by Codex, no rules.

Normal app	Recovery removed
PASS	PASS

The test never asked the save to fail.

Ask for rejection and recovery.

The rules tonight's command adds. 201 words in total.

Rules file, verbatim

Make the server reject the action (intercept the request, answer with an error). Assert the interface returns to its previous state and shows no success indicator.

3 runs per model

Rollback as the first failure: Codex 0/3 to 2/3, Claude 0/3 to 0/3. Claude caught the broken app 3/3 without the rules.

With the rules, Codex reached the rollback in 2 of 3 runs. Claude stopped at the alert.

Reach rejection before checking recovery.

Written by Codex, with rules.

1 Hold the POST, 3 Return a 500

```
if (route.request().method() !== 'POST')
  return route.continue();
postCount += 1;
await gate;
if (reject) {
  await route.fulfill({ status: 500,
    json: {
      error: 'Could not save like' } });
…
```

2 Open the gate, 4 Assert recovery

```
} finally {
  release();
}
await expect(button).toBeEnabled();
await expect(button).toHaveAttribute(
  'aria-pressed', String(!reject));
await expect(page.getByText(
  `${reject ? 128 : 129} likes`,
  { exact: true }
  )).toBeVisible();
```

Hold the response, release a rejection, then check recovery.

This assertion catches missing rollback.

Written by Codex, with rules.

Rejection branch

```
51  await expect(button).toHaveAttribute('aria-pressed', String(!reject));
```

Normal app	Recovery removed
PASS	**FAIL** Expected false, received true.

It failed while observing the removed rollback.

A red result can stop early.

Written by Codex, with rules. Rollback and error removed.

Rejection branch

```
49 if (rejected) {
50 await expect(page.getByRole('alert'))
     .toHaveText(
     "Couldn't save. Try again.");
51 await expect(button).toBeEnabled();
52 await expect(button).toHaveAttribute(
     'aria-pressed', 'false');
```

First failure

FAIL

Line 50, alert missing.
Lines 51 and 52 never ran.

An assertion that never ran observed nothing.

Another app. Earlier rules caught it.

TanStack Query example. Earlier study, 136 word rules.

Rollback deleted, no rules	Rollback deleted, with rules
PASS	**FAIL**
Claude 3/3, Codex 3/3. Rollback not checked.	Claude 3/3, Codex 3/3. Failed on the broken app.

FAIL is the result we want.

Change one thing. Check the first failure.

Step 1

Define the required evidence

Step 2

Remove one behavior

Step 3

Check why it fails first

Someone who didn’t write the test picks what to remove.

One scan finds the quiz line in under 10 seconds.

Cal.com at the commit before the fix. Scanner 3.5.3.

Scan output

```
Status: INCOMPLETE; confirmed P0: 17, P0 candidates: 31, unresolved locations: 401.
[P0][CONFIRMED] #4f Static finding (17 hits)
  "ab-tests-redirect.e2e.ts":76: #4f
  "out-of-office.e2e.ts":203: #4f
  "out-of-office.e2e.ts":370: #4f
```

Start with the confirmed ones; candidates need a human look.

Tonight: one habit, one command.

github.com/voidmatcha/e2e-skills

First, without any tool

Pick one test, remove the app behavior it checks: it must fail there.

Then scan one E2E folder (Node.js 22.20+)

```
npx -y --package=@voidmatcha/e2e-skills-scanner@3.5.3 e2e-skills \
  scan tests/ --format text --fail-on p0 --allow-incomplete
```

Later: add the rules, install the skills, GitHub Action `voidmatcha/e2e-skills@v3`

What protection remains?



Kent Beck

Rediscovered TDD

“deleting assertions from tests, deleting whole tests”



Kent C. Dodds

Creator of React Testing Library

“Green checks are not confidence”

Know what green protects.

Which condition did it reach?

Reached:

a successful save only.

What did its assertion observe?

Observed:

HTTP 200 and liked after reload.

Request: separate rejection coverage.

github.com/voidmatcha/e2e-skills

Thank you.



Yongjae Lee

Senior Front-End Engineer

github.com/voidmatcha/e2e-skills