9ème édition de la

Soirée du Test Logiciel Sophia - Antipolis

15 septembre 2026

17h à 23h

ETSI, Valbonne

Soirée du test logiciel

Sophia Antipolis

QA Prompt Engineering

Building Reliable AI Testing Workflows

Mohamed Hamza

THE SETUP

We all wrote this prompt at least once

YOUR PROMPT

“Write Playwright tests for the checkout conditional-offer feature.”

Eight words. Sent at 16:40 on a Friday.

WHAT CAME BACK

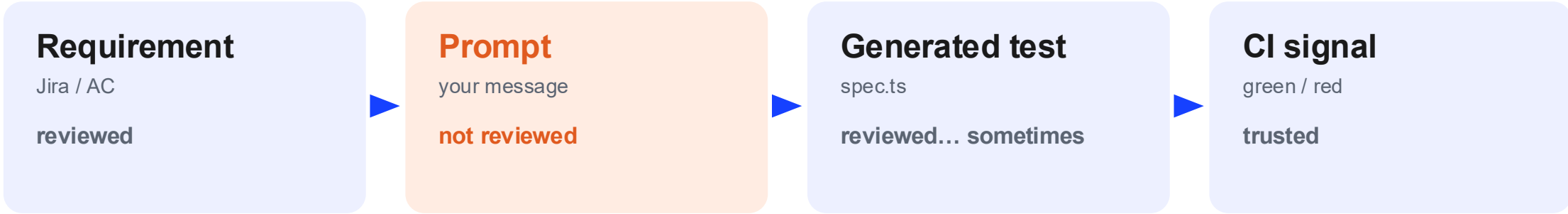
- A spec that compiles, runs green, and reviews “fine”
- page.waitForTimeout(3000) in three places !!
- A brand-new LoginPage, we already had one ☺
- productId = 123456 hardcoded in the spec
- expect(total).toBeVisible() instead of the amount toBeEqual()
- No teardown the offer leaks into the next run

It didn’t fail. It passed. That’s the expensive part.

A generated test that is green for the wrong reason is worse than no test it buys false confidence, at scale.

THE UNCOMFORTABLE PART

The prompt became the test specification



- **One link in that chain has no review step,**
  and it is the only artefact that gets reused fifty times this quarter.
- **Code defects are local. Prompt defects are systematic.**
  A weak prompt template ships the same flaw into every spec your team generates.
- **You already know how to handle an untested artefact.**
  Specify it. Test it. Version it. Regression-test it. It just happens to be prose.

WHY PROMPTS FAIL IN PRODUCTION

It is an information problem, not a wording problem

1 Missing context, not weak wording

The agent cannot know your data layer, your fixtures, your folder rules. Rewriting the sentence more politely does not close an information gap.

2 Under-specified assertions

“Verify the discount” gets you a visibility check. If you did not state the formula and the expected string, you did not test the calculation.

3 No reuse rule

An agent’s default action is to create. With no “search before you write” instruction you get a second LoginPage and a slowly forking page-object layer.

4 Conventions that live in a human head

File path, naming, tags, teardown, data ownership. If it was never written down, it is not in the prompt and it will not be in the output.

5 Snowflake prompts

Five engineers, five prompt styles, five quality levels. Output quality now tracks who asked, not what was asked. Unmeasurable, so unimprovable.

6 No teardown instruction

Generated tests happily create data and leave it. Run three, and test four fails for reasons that have nothing to do with the feature.

DEFINITION

What QA Prompt Engineering actually means

Treating the prompt as a versioned, testable artefact inside your test-automation supply chain.

Not a chat message. An input contract with a known-good output, an owner, and a regression suite.

1 **SPECIFY**
A fixed contract of context slots same shape, every request

2 **GENERATE**
Run it against the agent your team actually uses

3 **EVALUATE**
Score the output on dimensions, because you cannot diff it

4 **REGRESS**
Re-run a golden set on every template or model change

Prompt engineering, in this context, is context engineering.

The craft is not phrasing. It is deciding what the agent must be told to succeed — and then proving that list is complete. The loop above closes: what you learn in EVALUATE goes back into the contract.

DISCIPLINE 1 — SPECIFY

The prompt contract: slots, sections, not one paragraph (customisable)

Context
Business rules why a test should pass, not just the clicks

Environment
Base URL, store, user, fixtures determinism

Test data
Static, automation-owned products and IDs

APIs
Setup and teardown of state

Test cases
IDs, titles, exact ordered steps

Location
Exact path + reference specs convention by example

Assertions
Expected values and formulas. Highest-leverage slot

Resources
What exists vs. what may be created

Locators
The one thing the agent cannot infer

Guardrails
No hard waits, no branching mega-functions, no git ops

An empty slot is not a blank it is a decision you delegated to the agent by accident.

The constant slots (role, guardrails, house rules) do not belong in the prompt at all. Hoist them into the repo: AGENTS.md, Copilot instructions, a Claude Skill. Per-request prompts should carry only what is feature-specific.

DISCIPLINE 2 — EVALUATE

You cannot diff generated code. So score it.

Completeness

Is every requested case and assertion present?

Count ACs vs. tests greppable

Correctness

Does it pass for the right reason?

Break the feature on purpose; it must go red

Convention fit

Path, naming, imports, tags, structure?

Lint + the reviewer's diff

Reuse

Did it re-create something that existed?

Count new helpers that duplicate old ones

Determinism

Will it still be green on a slow night?

grep waitForTimeout, sleep, fixed indexes

Rework cost

How much did a human touch before the PR?

% of lines changed the number leads care about

Headline metric: rework rate. Target under ~10% of lines touched before the spec is PR-ready.

Non-determinism means you assert on properties, not on equality the same problem you already solved for flaky UI tests.

LIVE DEMO

From a ticket to a spec you would actually merge

LIVE DEMO

github.com/QA-MHA/JawdaPrompTest

WHAT TO WATCH FOR

- A UI to ease it for all types of profiles
- Which slots I fill fast, and which I refuse to skip
- The generated prompt itself: what the agent finally receives
- What is mandatory and what is optional to fill

TAKEAWAY

The delta came from context slots, not from better sentences.

Soirée du test logiciel

Conférence

FIELD NOTES

What actually moved the needle, and what did not

1 Structure beat wording, every time

The biggest single jump came from fixing the section order. Not from better adjectives.

2 Locators were the top accuracy lever

The cheapest thing a human can supply, and the one thing the agent cannot infer.

3 Negative rules outperformed positive ones

“Don’t use hard waits” moved code hygiene more than any amount of “write clean tests”.

4 Hoisting constants shrank the prompt

Moving role and rules into the repo cut prompt size and raised consistency across the team.

5 A plausible schema is not a validated template

Fill in a form and it runs. Validate against 1–3 real requests and it works. Those are different claims.

6 It raises the floor, not the ceiling

Your weakest prompt gets much better. Your best engineer gets slightly faster. Review before merge stays.

Soirée du test logiciel

Conférence

TAKEAWAYS

What to do tomorrow morning

- **Write your team's prompt contract on one page.**
  Your slots, sections, mandatory, optional.
- **Pick your three best real requests that is your golden set.**
  Store them with their validated output, next to the prompt.
- **Score on properties, and report rework rate.**
  Six dimensions for engineers. One number for everyone else.
- **Move the constant 40% into the repo.**
  AGENTS.md, Copilot instructions, or a Skill. Once, for everyone.
- **Industrialize | Create a process a UI with mandatory sections to fill**
  Promptest as example
- **Never merge a generated test you have not seen go red.**
  Create your review agents, skills, GitHub checks

IF YOU REMEMBER ONE THING

Your prompt is a test asset.

Specify it. Score it. Version it. Regression-test it.

Everything shown today runs on tooling you already have — the method is the transferable part.

Ex:

github.com/QA-MHA/JawdaPrompTest

9ème édition de la

Soirée du Test Logiciel Sophia - Antipolis

15 septembre 2026

17h à 23h

ETSI, Valbonne

Soirée du test logiciel

Sophia Antipolis

Merci de votre écoute !

SCAN ME

Votre avis nous intéresse