SauceLabs

Better Prompts, Better Tests

Your Guide to AI-Assisted Test Authoring

Better Prompts, Better Tests

There’s more code than ever out there. But none of that code is any good if it doesn’t pass quality tests. And that means the need to test it has never been higher.

A well-crafted test case prompt can generate a complete, accurate test flow in seconds. But a vague prompt produces tests that miss edge cases, target the wrong elements, or require heavy manual correction.

Think of prompting for test authoring the same way you'd think of writing a clear test plan: the more precise and structured your input, the more reliable and maintainable your output.

AN EFFECTIVE PROMPT FOLLOWS THIS STRUCTURE FOR MAXIMUM CLARITY AND RESULTS:

Specific Actions:

Use exact labels and verbs visible on the UI

Expected Outcomes:

Define what success looks like after each step

Structured Steps:

Break complex flows into discrete, numbered actions

Context:

Guide the model to hidden elements and dynamic content

Principle 1

Be Specific with Actions and Elements

AI models interpret natural language, but ambiguity leads to guesswork. Use the exact language visible on the UI.

SPECIFIC ACTIONS & ELEMENTS

Use exact button labels, field names, and page titles to anchor the model to what it actually sees.

Login flow

DON'T:
~~"Handle the login."~~

DO:
"Enter valid_user@example.com in the 'Email' field. Click the 'Sign In' button."

Navigation

DON'T:
~~"Go to settings and change the theme."~~

DO:
"Click the hamburger menu icon. Click 'Settings'. Under 'Appearance', select 'Dark Mode'."

Dynamic elements

DON'T:
~~"Click the trash can."~~

DO:
"Click the delete icon in the row containing 'Unnecessary Report'."

Rule of Thumb If a human reading your prompt couldn't identify the exact UI element without guessing, rewrite it.

Principle 2

Describe the Expected Outcome

Every meaningful test step should include a verifiable result. Without it, the AI can only confirm an action was taken — not that it worked.

EXPECTED OUTCOMES

Define what success looks like after key interactions.

Assertion checklist:

- ☑ What URL or page should load?
- ☑ What text or element should appear or disappear?
- ☑ What state should a field, toggle, or component be in?

Login flow

DON'T:	DO:
~~"Click 'Submit' after entering the required fields."~~	"Click 'Submit' and verify the page redirects to /dashboard and displays 'Registration Complete'."

Navigation

DON'T:	DO:
~~"Log out of the account."~~	"Click 'Log Out' in the user dropdown. Verify the page redirects to /login and the session cookie is cleared."

Pro Tip For assertions, specify: What URL should load? What text should appear? What state should a component be in?

Principle 3

Decompose Complex Flows

Long, multi-action prompts written as a single sentence are harder for AI to parse correctly. Break flows into discrete steps.

STRUCTURED STEPS

Break multi-screen workflows into sequential steps for better AI interpretation.

INSTEAD OF:

"Log in, find the product, add it to the cart, apply a promo code, and check out."

USE:

1. Navigate to /login. Enter testuser and Pass@word1. Click “Sign In”.

2. In the search bar, type "Wireless Headphones" and press Enter.

3. Click the first result titled "SoundPro 500".

4. Click “Add to Cart”. Verify the cart icon shows a count of 1.

5. Enter promo code SAVE20 and click “Apply”. Verify a $20 discount.

6. Click “Proceed to Payment”.

Pro Tip Think of each numbered step as a single user action with a clear before and after state.

Principles 4 & 5

Context & Anchoring

Guide the AI to hidden elements and use surrounding context to uniquely identify dynamic elements.

HIDDEN ELEMENTS & DYNAMIC CONTENT

The AI sees the rendered UI — not your source code. Guide it explicitly.

Hidden elements — guide the model:

"The 'Export' option is only visible after clicking the ':' icon in the top-right corner."

"The 'Advanced Filters' section is collapsed by default. Click 'Show Advanced Filters' to expand it."

"The date picker opens as an overlay modal. Select March 15 from the calendar grid."

Dynamic elements

DON'T:

~~"Click the edit icon."~~

DO:

"Click the 'Edit' icon in the row containing the user 'Jane Smith'."

Element specificity

DON'T:

~~"Click the first button."~~

DO:

"Click the 'Details' button in the row where the price shows $25.99."

Contextual anchoring

DON'T:

~~"Select the checkbox."~~

DO:

"Check the checkbox next to 'Enable two-factor authentication' in the Security section."

Pro Tip When elements don't have unique labels, use surrounding context like row content or section headers.

Principle 6

Leverage Structured Formats

AI test tools can interpret Gherkin, pseudo-code, and other structured test syntax and convert them into automated flows.

STRUCTURED FORMATS

AI test tools can interpret and convert structured test syntax into automated flows.

Gherkin Example

```
Feature: User Login

Scenario: Successful login
  Given I am on "/login"
  When I enter "valid_user@example.com" in the "Email" field
  And I enter "SecurePass1!" in the "Password" field
  And I click the "Sign In" button
  Then I should be redirected to "/dashboard"
  And I should see "Welcome back"
```

Pseudo-code Example

```
1. NAVIGATE to "/checkout"
2. FILL "First Name" with "Test"
3. FILL "Last Name" with "User"
4. SELECT "Visa" from "Card Type"
5. FILL "Card Number"
   with "4111111111111111"
6. CLICK "Place Order"
7. ASSERT URL contains
   "/order-confirmation"
8. ASSERT "Order #" is visible
```

Pro Tip If your team uses BDD or has existing test documentation, paste those formats directly!

Principle 7-10

Additional Principles

Test one scenario per prompt

Resist the urge to test everything in one prompt. Focused prompts produce cleaner, more maintainable test cases. Think of each prompt as a single test case, not a test suite.

DON'T:

~~"Test login, sign-up, forgot password, logout, and profile editing all in one flow."~~

DO:

"Test that a user with an expired subscription sees the upgrade banner on the dashboard."

Scope tests to the frontend

AI test authoring tools work by observing and interacting with the rendered UI. They do not have access to backend logic, APIs, or database state. Write prompts that describe what a user would see and do, not what your code does internally.

DON'T:

~~"Trigger the /api/users/create endpoint."~~

DO:

"Fill out the 'New User' form and click 'Save'. Verify the new user appears in the table."

Iterate and refine

If the AI takes a step you didn't want, stop the generation immediately and re-prompt. Don't let incorrect steps accumulate as they compound errors in subsequent steps. When re-prompting, apply the same specificity principles:

1. Say what went wrong and what you expected.
2. Redirect with a precise instruction rather than a vague correction.

EXAMPLE RE-PROMPT

"Stop. The previous step clicked the wrong 'Delete' button. Click the trash icon in the row containing 'Draft Report - Q1'."

Handle modals strategically

Most AI test tools handle common interruptions like cookie banners and browser permission dialogs automatically. For non-standard or mission-critical modals, provide explicit instructions.

- **Let the AI handle implicitly:** Cookie consent banners, generic "Got it" notifications, standard browser alerts.
- **Instruct explicitly:** Custom confirmation dialogs, multi-step modals, popups that block the next action and contain important choices.

EXAMPLE

"After clicking 'Delete Account' a confirmation dialog will appear. Click 'Confirm Delete' in that dialog."

Prompting with Sauce AI for Test Authoring

Sauce AI for Test Authoring is a Sauce Labs feature that allows you to generate automated test flows using natural language prompts directly within the Sauce Labs platform. It analyzes the live rendered state of your app to identify UI elements and generate step-by-step test interactions.

Key Sauce AI behaviors to know

Sauce AI sees only the rendered frontend.

It cannot access your application's source code, API calls, or backend state. Always describe what is visible in the UI.

Structured prompts are supported.

Gherkin, pseudo-code, and numbered steps can be pasted directly and converted to automated flows.

Common obstacles are handled automatically.

Cookie consent popups, standard browser dialogs, and similar interruptions are typically managed without explicit instructions. Reserve explicit handling instructions for non-standard, complex modals.

Guidelines

Specific guidelines for getting the best results from Sauce AI for Test Authoring.

Avoid vague language — use labels and verbs

AI models interpret natural language, but ambiguity leads to guesswork. Use exact button labels, field names, page titles, and verbs visible on the UI.

DON'T:

~~"Handle the login."~~

DO:

"Navigate to the User Profile page. Enter valid_username in the Username field. Click 'Sign In'."

Re-prompt early and precisely

If Sauce AI takes an unintended step, stop the generation and redirect it using specific language. Avoid letting incorrect steps continue.

EXAMPLE:

"Stop. That clicked the wrong button. Click the 'Edit' icon in the row for 'Project Alpha', not 'Project Beta'."

Guide model to hidden elements

If a navigation item, button, or section is not visible until a trigger is activated, tell Sauce AI how to reveal it first.

EXAMPLE:

"The 'Reports' section is hidden under the 'Analytics' submenu. Click 'Analytics' in the left sidebar to expand the submenu, then click 'Reports'."

Specify expected outcomes for assertions

Sauce AI needs to know what a successful step looks like. Always define the expected state after key interactions.

EXAMPLE RE-PROMPT

"Click the 'Submit' button and verify that the page redirects to /dashboard and displays the success message: 'Registration Complete'."

Use context to clarify repeated elements

For lists, tables, or repeated components with identical icons or labels, use row content or surrounding text to uniquely identify the element.

EXAMPLE

"Click the delete (trash can) icon for the 'Unnecessary Report' item." "Click the 'Details' button in the row containing the price $25.99." "Click the 'Edit' icon next to the most recently added user."

Examples

Prompt Examples by Scenario

Real-world examples showing how to prompt Sauce AI for common test scenarios.

Use these as templates for your own test authoring prompts.

Authentication Flow

1. Navigate to "/login".
2. Enter "qa_user@saucelabs.com" in "Email".
3. Enter "TestPass123!" in "Password".
4. Click "Sign In".
5. Verify redirect to "/home" and user avatar appears.

E-Commerce Cart

1. Search for "Bluetooth Speaker".
2. Click the first result.
3. Select "Blue" from "Color" dropdown.
4. Click "Add to Cart".
5. Verify cart shows quantity of 1.

Form Validation

1. Navigate to "/register".
2. Leave "Email" field empty.
3. Enter "weak" in "Password".
4. Click "Create Account".
5. Verify error messages appear.

Deletion with Modal

1. Navigate to "Users" page.
2. Locate row for "Test User".
3. Click trash icon in that row.
4. Click "Delete" in confirmation modal.
5. Verify user no longer appears.

Quick Reference

Prompt Checklist

Before submitting a prompt, run through this checklist to ensure maximum effectiveness.

Run through these checks before submitting any test authoring prompt.

Specific elements

Are you using the exact text label, field name, or button text visible on the UI?

Clear actions

Are your verbs unambiguous? (Click, Enter, Select, Verify, Navigate)

Expected outcomes

Does each key step include what success looks like?

Context for hidden elements

Have you guided the model through menus, drawers, or modals needed to reach the target?

Disambiguation

If there are repeated elements, have you used row content or surrounding text to identify the right one?

Scoped to the UI

Is your prompt describing what a user sees and does, not backend or API behavior?

GLOSSARY

Assertion / Verify	A test step that checks an expected condition — e.g., text is visible, URL has changed, element is disabled.
Gherkin	A structured, plain-English syntax (Given/When/Then) used in BDD frameworks like Cucumber. Supported by Sauce AI.
Dynamic Element	A UI element without a fixed, unique label — such as icons repeated in a list. Requires contextual anchoring in prompts.
Modal	An overlay dialog that appears on top of the page, typically requiring user action before proceeding.
Pseudo-code	Simplified, human-readable code-like instructions. Useful for describing test flows in a structured format.
Re-prompting	Correcting an AI's behavior mid-generation by stopping and submitting a more precise instruction.

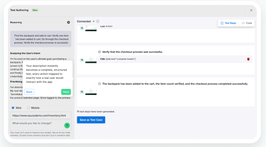
Takeaways

The efficacy of AI-powered test authoring hinges entirely on the quality of your input.

By embracing universal principles — specificity in actions, clarity in assertions, and structure in flow — you transform a potential black box into a powerful, automated partner. Users of Sauce AI for Test Authoring should leverage contextual anchors and precise, descriptive language to ensure faster, more accurate test generation. Adopt these best practices, iterate relentlessly, and you will unlock the true potential of AI to create robust, maintainable test suites at the speed of development.

Sauce AI product documentation

Visit docs.saucelabs.com/sauce-ai/ai-authoring/



See Sauce AI for Test Authoring in action

Take the interactive product tour.

About Sauce Labs

Schedule a conversation

Sauce Labs offers a platform for continuous quality that supports software teams across the entire software development lifecycle. The unified platform enables continuous quality using AI-driven analytics to identify key quality signals from development through production. With deep roots in the Selenium and Appium open-source communities, and best-of-breed infrastructure for automated and manual testing of web and mobile applications, Sauce Labs helps teams test across thousands of different devices, browsers, and operating systems at any scale.

- VIRTUAL DEVICE CLOUD
- REAL DEVICE CLOUD
- VISUAL & ACCESSIBILITY
- MOBILE APP DISTRIBUTION
- CRASH & ERROR REPORTING
- ADVISORY SERVICES