QA TEST PLAN

SauceDemo E-Commerce Platform

Senior QA Engineer Challenge

Version 1.0 | February 2026

---

1. Scope and Objectives

1.1 Objectives

This test plan defines the strategy for validating the critical functionality of the SauceDemo e-commerce platform. The primary goal is to establish a reliable smoke test suite that acts as a CI/CD gate — if any of these tests fail, a deployment should not proceed.

1.2 In Scope

- End-to-end happy path: Login → product selection (random) → cart management (add and remove items) → checkout completion
- Login functionality: successful authentication with standard_user
- Login negative scenarios: multiple failed login attempts and expected system behavior
- Post-checkout validation: confirmation message and cart reset to empty state
- API security validations: login payload inspection, protected endpoint access control, cart state persistence

1.3 Out of Scope

- Alternative user accounts (locked_out_user, problem_user, performance_glitch_user, error_user, visual_user) — only standard_user is in scope for this challenge
- Known UI bugs documented publicly (visual regressions, layout issues)
- Performance and load testing
- Accessibility (a11y) testing
- Cross-device / mobile responsive testing
- Payment gateway integration (SauceDemo uses a simulated checkout with no real transactions)

---

2. Test Approach

2.1 Functional Testing Strategy

Testing will focus on the critical user journey (happy path) and key negative scenarios. All

automated tests will be implemented using Playwright with TypeScript, following a Page Object Model (POM) architecture to ensure maintainability and reusability.

Two layers of testing will be implemented:

- Smoke / E2E tests: validate the complete purchase flow from login to order confirmation
- API security tests: validate request payloads and endpoint access control using Playwright's request context

2.2 UI Validation Approach

- Assertions will target semantic selectors (data-test attributes, roles) over fragile CSS classes
- Key UI checkpoints: login page renders correctly, products page loads after auth, cart badge updates on add/remove, confirmation message displays after checkout, cart is empty after successful purchase, generic error message displays on failed login (note: no account lockout message appears after multiple failed attempts — expected lockout behavior is absent and documented as a security risk)

2.3 API Validation Approach

Although SauceDemo is a frontend-only demo, Playwright's request context will be used to inspect network behavior during login and navigation. The following will be validated:

- Login request payload does not expose the password in plaintext
- Accessing protected routes without a valid session returns HTTP 401 Unauthorized
- Cart state persists in session storage across page refreshes (session behavior)

---

3. Risk Assessment

Risk	Severity	Mitigation
No account lockout after multiple failed login attempts (brute force vulnerability)	High	Documented as a security finding. standard_user does not get blocked — no lockout message appears. Would be treated as a critical defect in a production environment.
Password transmitted in plaintext in login request payload	High	Include as an explicit API assertion. Fail the test if password is visible in the request payload.
Protected routes accessible without authentication (missing 401)	High	Add API test to verify access control on at least one authenticated endpoint.
Demo app instability / downtime (external dependency)	Low	Tests are designed to fail gracefully with descriptive error messages. No mitigation for third-party uptime.

3.1 Considered and Excluded

- Cart state after logout: Cart contents persist after logout — this is expected, non-blocking behavior and does not affect the purchase flow. Excluded from the CI gate scope.
- Random product selection picking a sold-out item: Not applicable for standard_user. All products are available. Considered and excluded from scope.

---

4. Entry and Exit Criteria

4.1 Entry Criteria

- SauceDemo application is accessible at https://www.saucedemo.com/
- Test credentials (standard_user / secret_sauce) are valid and functional
- Playwright project is set up with TypeScript and all dependencies installed
- All POM classes are implemented and the project compiles without errors

4.2 Exit Criteria

- All smoke test scenarios pass across Chromium, Firefox, and WebKit
- All API security assertions pass with no plaintext credentials detected
- Test execution report is generated with 0 critical failures
- Any identified security findings are documented as known issues

---

5. Environment Requirements

5.1 Browser Support

For the CI/CD smoke gate (run on every commit), tests will execute on Chromium only. This prioritizes speed and keeps the feedback loop short for developers. Cross-browser coverage (Firefox and WebKit) is configured as a separate Playwright project matrix, intended for scheduled runs or pre-release validation — not the commit gate.

- Smoke gate (per commit): Chromium
- Extended suite (scheduled / pre-release): Chromium, Firefox, WebKit

5.2 Test Data

- Primary user: standard_user / secret_sauce
- Product selection: randomized at runtime from the available product list on the inventory

page
- Checkout form data: static test data (e.g., First Name: Test, Last Name: User, Zip: 10001)
- No database seeding required — SauceDemo resets state can be done manually.

5.3 Tooling

- Test framework: Playwright 1.x
- Language: TypeScript
- Design pattern: Page Object Model (POM)
- Node.js: v18 or higher
- Package manager: npm
- CI readiness: tests executable via a single npm run test command

---

6. Assumptions & Limitations

- SauceDemo is a demo application. Security findings (e.g., no brute force protection) are documented as observations and would be treated as critical defects in a production environment.
- The checkout flow does not process real payments. The completed purchase is simulated.
- The API test targets network-level behavior observable through the browser. No dedicated backend API documentation is available for this demo.
- Cart persistence is tested via browser session behavior, not a backend API endpoint.