TEST PLAN & TEST STRATEGY DOCUMENT

Project: SauceDemo UI Automation
Application Under Test (AUT): https://www.saucedemo.com
Prepared For: QA Automation Practice
Prepared By: Nadja Celik-Salcinovic
Version: 1.1
Date: 17.01.2026

---

1. Introduction

This document defines the overall test plan and test strategy for validating the functionality, quality, and reliability of the SauceDemo web application. It outlines testing scope, objectives, environments, tools, and test types. The plan applies to manual and automated testing, with an emphasis on UI automation using **Playwright**.

---

2. Objectives

The primary objectives of this test effort are:

1. Validate that core user workflows (authentication, product browsing, cart management, checkout, order completion) function correctly.
2. Ensure UI elements render correctly under various conditions.
3. Establish a maintainable and scalable automation framework using **Playwright**.
4. Identify defects early and ensure stable release readiness.
5. Provide transparent reporting on coverage, defect trends, and test results.

---

3. Scope

3.1 In Scope

- Web UI testing on **Chrome** (primary)
- Functional testing of all key flows:
    - Login/logout
    - Product browsing
    - Sorting & filtering
    - Add/remove items
    - Cart operations
    - Checkout flow (Step 1, Step 2, Completion)
    - Side menu operations (About, Reset App State, Logout)

- Validation of core UI components: buttons, forms, dynamic data
- End-to-end automation using **Playwright**
- Data-driven testing using available credentials

3.2 Out of Scope

- Performance testing
- Security testing (SQL injection, XSS, etc.)
- Backend API testing
- Mobile testing on real devices
- Load testing
- Accessibility audits (WCAG compliance)

---

4. Test Approach / Test Strategy

4.1 Testing Levels

1. **Smoke Testing**
    - Validate that the application is operational and major workflows run without blockers.
2. **Functional Testing**
    - Validation of all user-critical features using manual and automated tests.
3. **Exploratory Testing**
    - Assess edge cases, unexpected behavior, and user-experience aspects.

- **4.2 Automation Strategy (Rewritten for Playwright)**

Framework: Playwright v1.x+ for UI automation.

Browsers: Chrome, Firefox, and WebKit (cross-browser coverage).

Test Structure:

Use Page Object Model (POM) to separate page logic from tests.

Group tests using test.describe() for logical modules:

Login / Logout

Products / Sorting / Filtering

Cart Operations

Checkout Flow

Menu Operations (About, Reset App State, Logout)

Hooks & Fixtures:

test.beforeEach() to perform login or common setup.

test.afterEach() to capture screenshots on failure and reset state.

Selectors:

Use stable data-test attributes from SauceDemo for robustness.

Avoid brittle CSS/XPath selectors that break on UI changes.

Synchronization & Flakiness Handling:

Playwright handles waits automatically, but explicit waits can be used for animations.

Retry failed tests up to 1–2 times in CI to reduce flakiness.

Test Types:

Smoke tests → 5–10 high-value flows across browsers.

Functional regression → covers all core features.

Negative / edge case tests → invalid logins, empty cart, checkout errors.

Cross-browser validations → ensure layout, UI, and core flows work in Chrome, Firefox, and WebKit.

Data-driven Testing:

Use Playwright fixtures or test parameterization to run tests with multiple user types (standard, problem, locked-out, performance glitch).

CI Integration:

GitHub Actions pipeline to run tests in headless and headed mode on multiple browsers.

Capture artifacts (screenshots, videos, logs) on failures.

Reporting:

Generate HTML and JSON reports for each test run.

Track test coverage per module and per browser.

4.3 Test Types

1. Functional Tests
2. End-to-End Tests
3. Negative Tests
4. Boundary Tests
5. UI Validation Tests

6. Cross-browser Smoke Tests

---

5. Test Deliverables

- Test Plan (this document)
- Test Scenarios & Test Cases
- Test Execution Reports
- Defect Reports in Issue Tracker
- Test Summary Report

---

6. Entry & Exit Criteria

6.1 Entry Criteria

- AUT is deployed and accessible
- Test data prepared (user credentials)
- Test environment stable
- Requirements/test scenarios reviewed

6.2 Exit Criteria

- All critical test cases executed
- Zero Critical and High defects open
- Test summary report delivered
- Automation coverage > 80% for core flows

---

7. Test Environment

7.1 Environment Setup

- Browser: Chrome (latest)
- OS: Windows 10/11
- Automation: **Playwright v1.x+**
- Node.js LTS version
- CI: GitHub

7.2 Test Data

- Valid user: standard_user / secret_sauce
- Problem user, locked-out user, performance glitch user

- Mocked test data for cart actions

---

8. Test Execution Strategy

- Smoke tests executed before each full regression
- Regression suite executed on major changes
- Failed tests analyzed with screenshots & logs

---

9. Risk & Mitigation

Risk	Mitigation
UI changes	Use stable selectors (data-test)
Flaky tests	Explicit waits avoided, retries used
Environment instability	Clean state before each test

---

10. Test Schedule

Phase	Duration
Test Design	2 days
Automation Development	5 days
Test Execution	2 days
Reporting	1 day

---

11. Defect Management

- Defects categorized by severity (Critical–Low)
- Repro steps, screenshots & logs attached
- Retesting after fix

---

12. Test Summary & Reporting

- Execution status summary
- Passed / Failed / Blocked metrics
- Coverage per module
- Lessons learned & improvement areas