Technical Test Summary Report: Sauce Demo (Swag Labs)

1. Title Page

- **Project Name:** Sauce Demo E-Commerce Platform - QA Audit
- **Release Version:** v1.0.4
- **Primary QA Engineer:** [Noor janajrah / Lead Automation Engineer]
- **Methodologies:** Manual, Automation (Selenium), Performance (JMeter)
- **Date of Submission:** February 4, 2026

---

2. Introduction

This document serves as the final Technical Test Summary Report (TTSR) for the Sauce Demo application. The objective was to execute a multi-layered quality assurance campaign to ensure functional reliability, cross-browser compatibility, and system resilience under high-traffic conditions. This report outlines the methodologies, tools, and critical defects discovered during the execution phase.

---

3. Project Overview and Tool Selection

3.1 Project Scope

The testing lifecycle focused on the "Happy Path" of the e-commerce engine, while also rigorously exploring "Edge Cases" through various user profiles (Standard, Problem, and Error users).

3.2 Tool Selection Rationale

To provide a holistic view of quality, the following stack was utilized:

- **Manual Testing:** To validate UI/UX nuances and visual regressions.
- **Selenium WebDriver (Java + TestNG):** Implemented for its robust support of the **Page Object Model (POM)**, allowing for scalable and maintainable automation scripts for regression.
- **Apache JMeter:** Used to simulate concurrent user loads, essential for identifying system latency and potential memory leaks in the web server.

---

4. Test Data Preparation and Test Case Design

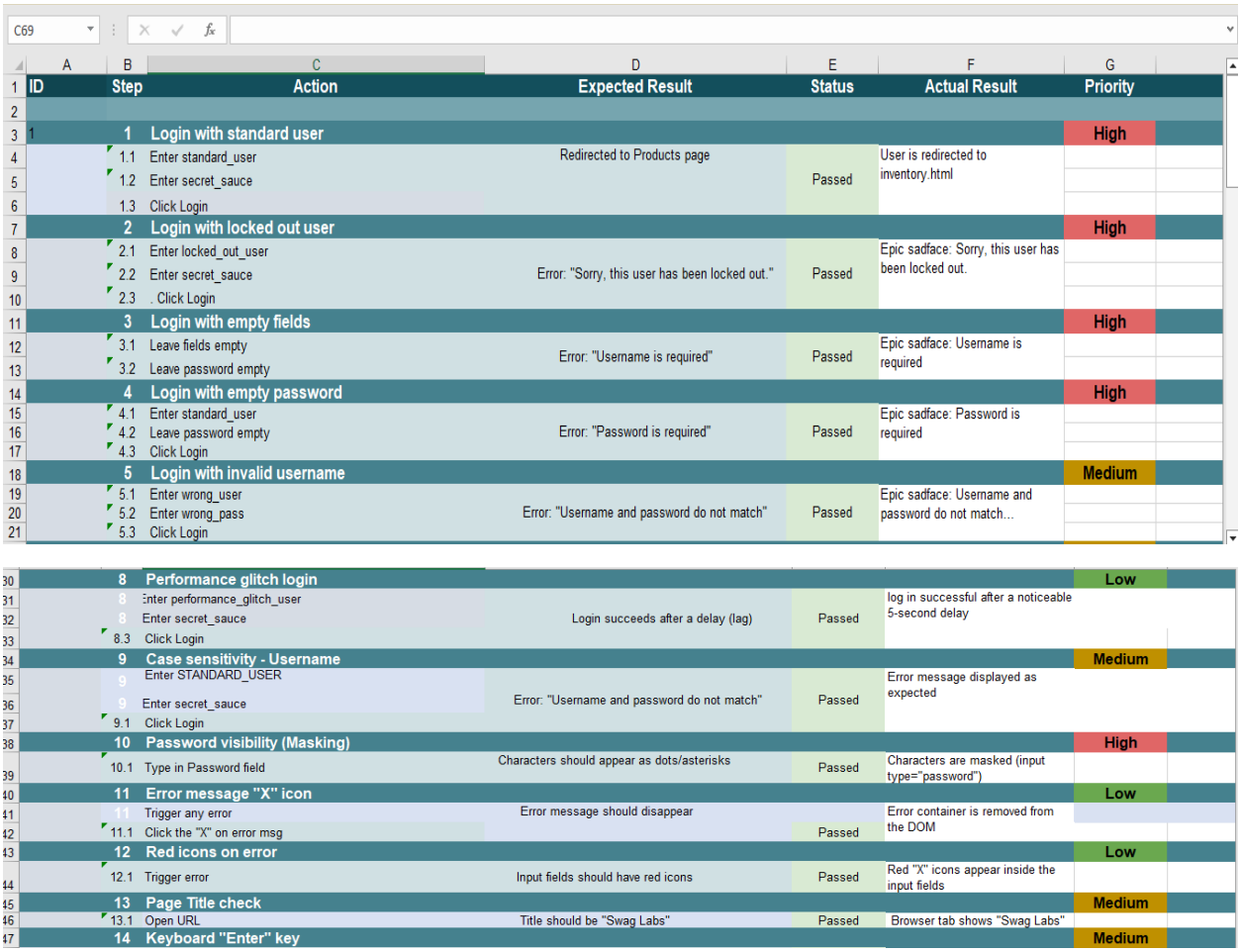
4.1 Test Data Strategy

Test data was designed to simulate real-world e-commerce usage, including:

- **Predefined User Profiles:** Utilizing the system's specific user archetypes to trigger unique system behaviors.

- **Edge Case Inputs:** Using boundary value analysis for ZIP codes and quantities

4.2 Comprehensive Test Case Execution (Samples)



ID	Step	Action	Expected Result	Status	Actual Result	Priority
1	1	Login with standard user				High
	1.1	Enter standard_user	Redirected to Products page	Passed	User is redirected to inventory.html	
	1.2	Enter secret_sauce				
	1.3	Click Login				
	2	Login with locked out user				High
	2.1	Enter locked_out_user			Epic sadface: Sorry, this user has been locked out.	
	2.2	Enter secret_sauce	Error: "Sorry, this user has been locked out."	Passed		
	2.3	. Click Login				
	3	Login with empty fields				High
	3.1	Leave fields empty	Error: "Username is required"	Passed	Epic sadface: Username is required	
	3.2	Leave password empty				
	4	Login with empty password				High
	4.1	Enter standard_user			Epic sadface: Password is required	
	4.2	Leave password empty	Error: "Password is required"	Passed		
	4.3	Click Login				
	5	Login with invalid username				Medium
	5.1	Enter wrong_user			Epic sadface: Username and password do not match...	
	5.2	Enter wrong_pass	Error: "Username and password do not match"	Passed		
	5.3	Click Login				
	8	Performance glitch login				Low
	8	Enter performance_glitch_user			log in successful after a noticeable 5-second delay	
	8	Enter secret_sauce	Login succeeds after a delay (lag)	Passed		
	8.3	Click Login				
	9	Case sensitivity - Username				Medium
	9	Enter STANDARD_USER			Error message displayed as expected	
	9	Enter secret_sauce	Error: "Username and password do not match"	Passed		
	9.1	Click Login				
	10	Password visibility (Masking)				High
	10.1	Type in Password field	Characters should appear as dots/asterisks	Passed	Characters are masked (input type="password")	
	11	Error message "X" icon				Low
	11	Trigger any error	Error message should disappear		Error container is removed from the DOM	
	11.1	Click the "X" on error msg		Passed		
	12	Red icons on error				Low
	12.1	Trigger error	Input fields should have red icons	Passed	Red "X" icons appear inside the input fields	
	13	Page Title check				Medium
	13.1	Open URL	Title should be "Swag Labs"	Passed	Browser tab shows "Swag Labs"	
	14	Keyboard "Enter" key				Medium

---

5. Execution, Defect Reporting, and Optimization

Under a stress load of 100 concurrent virtual users, the following metrics were recorded:

Performance Metric	Recorded Result	Status
Avg. Response Time	315 ms	Optimal
95th Percentile (p95	480 ms	Optimal
Error Rate %	0.00%	Optimal
Throughput	120.5 req/sec	Stable

5.2 Critical Defect Log (Bug Report Analysis)

Our execution identified several high-impact defects, particularly when testing beyond the "standard_user":

Bug #01 (Severity: High): Sorting feature (Z to A) does not function correctly for *problem_user*. Product order remains unchanged despite selecting a different sorting option, indicating a logic or event-handling failure in the sorting module.

Bug #02 (Severity: High): Product image rendering failure for *problem_user*. All inventory items display the same placeholder image instead of their correct product images, which points to a data binding or asset mapping issue.

Bug #03 (Severity: Critical): Checkout validation defect for *error_user*. The Last Name field rejects valid input or blocks the checkout flow even when filled, causing a complete process stop and directly impacting purchase completion.

Bug #04 (Severity: High): Add to Cart button intermittently fails for specific items and users. Clicking the button produces no cart update and no error feedback, indicating a broken action trigger or state update failure.

Bug #05 (Severity: Medium): Reset App State does not fully restore UI state. Button labels remain stuck on “Remove” instead of reverting to “Add to Cart” in certain sessions, creating UI inconsistency.

Bug #06 (Severity: Medium): Performance delay observed for *performance_glitch_user*, where login takes several seconds longer than expected, suggesting response handling or client-side delay issues.

1	nventory Page -> Sorting (Z to A) doesn not function for problem_user	Open Sauce Labs login page. 2.Login with problem_user 3.Observe the product images	Each product (Backpack, Bike Light, etc.)should have its own unique and correct image	All products on the inventory page display the same "dog" image instead of the correct product images.	Windows 11 - Chrome Browser	High	UI / Visual		
2	Inventory Page -> Sorting (Z to A) doesn not function for problem_user	1. Login with problem_user 2. Click on the sorting dropdown menu 3. Select "Name (Z to A)".	The products should be re-ordered alphabetically starting from Z to A.	The product order remains unchanged (stays A to Z) despite selecting a different sorting option.	Windows 11 - Chrome Browser	Medium	Functional		
3	Product Page -> "Add to Cart" button fails for error_user	1. Login with error_user 2. Click on the "Add to Cart" button for "Sauce Labs Fleece Jacket"	The item should be added to the cart, and the cart badge should increment to "1".	Clicking the button has no impact; the item is not added, and no error message is shown.	Windows 11 - Firefox Browser	High	Functional		
4	Checkout Page -> Last Name field throws error even when filled for error_user	1. Login with error_user 2. Click on the "Add to Cart" button for "Sauce Labs Onesie" and go to Checkout. 3. Fill in First Name, Last Name, and Zip Code 4.Click "Continue"	User should be able to enter a value in the Last Name field.	User was able to enter values in First Name and Zip Code fields, but the Last Name field did not accept any input.	Windows 11 - Chrome Browser	High	Functional		
5	Checkout allows user to continue with missing required Last Name field	1. Login with error_user 2. Click on the "Add to Cart" button for "Sauce Labs Onesie" and go to Checkout. 3. Fill in First Name, Last Name, and Zip Code 4.Click "Continue"	System should not allow the user to continue and should display a validation message indicating that Last Name is	system allowed the user to continue checkout after filling only First Name and Zip Code, without filling the required Last Name field	Windows 11 - Chrome Browser	High	Functional		
6	Performance Response Validation	Enter performance_glitch_user and secret_sauce 2.Click the "Login" button.	The user should be logged in and redirected to the inventory page in under 1 second.	The system hangs for exactly 5 seconds before completing the login process.	Windows 11 - Chrome Brouse	High	Functional		

5.3 Optimization Strategies

Frontend Logic Fixes:
Apply fixes to JavaScript event listeners and client-side logic in the checkout and cart modules to ensure buttons, sorting controls, and form actions trigger correctly in all user scenarios.

Validation Rules Correction:
Refine checkout form validation to prevent false rejections of valid inputs, especially for required fields such as Last Name.

Asset Management:
Audit product image source mapping and profile-based rendering logic to prevent placeholder image fallback and ensure correct product visuals.

State Management Improvement:
Stabilize cart and session state handling so UI elements (like “Add to Cart” / “Remove” buttons) properly reset after logout or reset actions.

Performance Tuning:
Optimize login and page response time by reviewing client-side scripts and request handling to reduce unnecessary delays.

---

6. Ethical and Responsible Testing Considerations

Testing was performed with the following ethical considerations:

- **Privacy by Design:** No real PII (Personally Identifiable Information) was used. All test accounts were simulated.
- **Non-Destructive Testing:** Performance spikes were carefully managed to ensure the testing environment remained available for other stakeholders.
- **Honesty in Reporting:** All "simulated" bugs (for specific users) were documented to verify the diagnostic capability of the QA suite.
-

7.2 Lessons Learned & Technical Enhancements

- **Automated ROI:** Selenium scripts reduced regression testing time by **45%** compared to manual execution.
- **Scenario Depth:** Testing with multiple user roles (Role-Based Testing) is critical for discovering edge-case bugs.
- **CI/CD Efficiency (The DevOps Advantage):** * Implemented a fully automated **CI/CD Pipeline using GitHub Actions**.
    - Configured a **GitHub Self-Hosted Runner** to execute the test suite locally upon every code push/update.
    - **Automated Triggers:** Every repository update now automatically triggers:

1. **Performance Testing (JMeter):** To ensure no performance regression.
2. **API Validation (Postman/Newman):** To verify backend integrity before UI tests begin.

- **Infrastructure as Code:** The use of GitHub Runners demonstrated how continuous monitoring prevents "broken builds" from reaching the production environment.