Test Plan for SauceLabs Demo Website

# 1. Introduction

This document details the test plan for the SauceLabs demo website (https://www.saucedemo.com/v1/index.html), focusing on functional validation of core features and user experience. This test plan is designed to identify potential issues in primary workflows and ensure the application's functionality aligns with user expectations.

# 2. Test Scope and Objectives

The primary objective of this test plan is to verify the functional integrity and usability of the SauceLabs demo website. The scope includes critical user workflows, such as login, product browsing, cart management, and checkout. The goal is to identify potential bugs, validate expected functionalities, and improve overall user experience.

# 3. Test Scope and Objectives

This section defines the overall approach to testing, including:

## 3.1 Types of Testing

- Functional Testing  
 - Regression Testing  
 - Performance Testing  
 - Usability Testing

## 3.2 Testing Levels

- Unit Testing  
 - Integration Testing  
 - System Testing  
 - User Acceptance Testing

# 4. Types of Functional Tests

4.1 Input Validation  
 - Validate login input fields with valid and invalid credentials.  
 - Test input boundaries, such as minimum and maximum character lengths for usernames and passwords.

4.2 User Flow Testing  
 - Validate end-to-end workflows for typical user actions, such as login, product selection, adding items to cart, and checkout to confirm these processes function correctly.

4.3 Error Handling  
 - Verify that appropriate error messages are displayed for invalid logins, empty cart checkout, and other common issues.

4.4 Boundary Testing  
 - Test limits of valid inputs and validate how the application handles unexpected inputs, such as special characters.

# 5. Testing Approach

This approach ensures coverage of critical functionalities and user flows, focusing on areas with high user interaction.

The tests are designed to quickly identify major defects in login, cart management, and checkout, which are essential for a functional e-commerce experience. Additionally, boundary and error handling tests validate application resilience to unexpected inputs.

# 6. Automation: Tools and Frameworks

To ensure efficient and reliable testing, the following tools and frameworks are suggested:

**Selenium WebDriver:** For cross-browser functional automation testing.  
**TestNG or JUnit:** For test structure and reporting.  
**Allure for Reporting:** Provides detailed test reports, including screenshots for failures.  
These tools offer flexibility, strong browser support, and reporting capabilities, making them ideal for this functional testing project.

# 7. Issue Reporting

During testing, any bugs or unexpected behaviors should be documented as follows:

**Steps to Reproduce:** Outline steps leading to the issue.  
**Expected Result:** Describe the intended functionality.  
**Actual Result:** Document what actually happened.  
**Impact:** Note the impact on user experience or application stability.

# 8. Test Environment

The test environment should mirror the production environment to ensure accurate test results. It will include:  
  
**Web Browser Requirements:** Chrome, Firefox, Edge (latest versions).  
**Operating System:** Windows 10/11, macOS.  
**Network Configuration**: Stable internet connection with typical latency settings to mimic end-user conditions.  
**Test Data:** Data sets covering all input variations and edge cases.

# 9. Deliverables

The following are deliverables to be provided at various stages of the testing lifecycle:  
  
**Test Plan Document:** This document outlining the test approach and scope.  
**Test Cases:** Detailed test cases covering functional, regression, and usability tests.  
**Test Scripts:** Automated scripts written for high-priority test cases.  
**Test Reports:** Summary of test execution, including defect density, pass/fail rates, and key metrics.  
**Defect Log:** Document of reported bugs with details on severity, priority, and status.

# 10. Entry and Exit Criteria

Entry Criteria:  
- Test environment setup is complete and verified.  
- Required test data is available and validated.  
- All necessary software builds are deployed in the test environment.  
- Test plan and test cases are reviewed and approved.  
  
Exit Criteria:  
- All critical test cases have been executed and passed.  
- High-priority defects have been resolved or accepted by stakeholders.  
- Test completion report has been reviewed and signed off.

# 11. Risk Analysis and Mitigation

Identified Risks:  
- **Insufficient Test Data:** Test data may not cover all edge cases.  
**Mitigation:** Define comprehensive test data with known boundary values and negative test scenarios.  
  
- **Delays in Test Environment Setup:** Delays can impact testing timelines.  
**Mitigation**: Schedule environment setup early in the project timeline and maintain backup environments.

- **Unidentified Defects in Regression:** New updates may introduce defects.  
**Mitigation:** Use regression test automation and prioritize high-impact areas.