# Test Plan: Sauce Labs Demo E-commerce Website

**Project:** saucedemo-automation-framework

**Author:** Tanish

**Version:** 1.0

## 1. Introduction

### 1.1. Overview

This document outlines the test plan for the Sauce Labs Demo e-commerce website, also known as "Swag Labs". The primary objective of this test effort is to ensure the quality, functionality, and reliability of the application's critical user flows through automated testing. This plan details the scope, approach, resources, and schedule of the testing activities.

### 1.2. Goal

* To verify that all critical functionalities of the application meet the expected requirements.
* To identify and report any bugs or defects in the application.
* To build a robust and maintainable regression test suite using Selenium and Pytest.
* To ensure the application provides a consistent user experience across major web browsers.

## 2. Scope

### 2.1. In-Scope Functionality

The following features and functionalities will be tested:

* **User Authentication:**
  + Login with valid credentials.
  + Login with invalid credentials.
  + Error message validation.
* **Product Inventory:**
  + View product listings.
* **Shopping Cart Management:**
  + Add single and multiple items to the cart.
  + Verify cart contents.

### 2.2. Out-of-Scope Functionality

The following will not be tested in this phase:

* Performance, load, or stress testing.
* Security vulnerability testing.
* API-level testing (current focus is UI automation).
* The checkout process beyond cart verification.
* The "About" page and other static content pages.

## 3. Test Strategy

### 3.1. Automation Approach

The primary approach is to use UI test automation to simulate user interactions. The framework is built using the **Page Object Model (POM)** design pattern to ensure scalability and maintainability.

### 3.2. Testing Levels

* **End-to-End Testing:** Test cases will be designed to cover complete user flows from login to cart verification.

### 3.3. Types of Testing

* **Functional Testing:** Verifying that the features work according to requirements.
* **Regression Testing:** Ensuring that new code changes do not break existing functionality. The automated suite will serve as the regression pack.
* **Cross-Browser Compatibility Testing:** Ensuring the application functions correctly on supported browsers.

## 4. Resources

### 4.1. Tools & Technology

* **Automation Tool:** Selenium WebDriver
* **Programming Language:** Python
* **Test Framework:** Pytest
* **Reporting:** Allure Report
* **Version Control:** Git / GitHub
* **CI/CD:** GitHub Actions
* **Project Management:** Jira

### 4.2. Test Environment

* **Operating System:** Windows, macOS, Linux (via GitHub Actions)
* **Browsers:**
  + Google Chrome (latest version)
  + Mozilla Firefox (latest version)

## 5. Test Deliverables

* **Test Plan:** This document.
* **Test Cases:** A detailed document outlining manual test cases.
* **Automation Scripts:** The complete Python test automation suite hosted on GitHub.
* **Test Execution Reports:** Allure reports generated after each test run.
* **Bug Reports:** (If applicable) Defects logged in Jira.