SwagLabs Test Plan

Version 0.1

Author: Howard Davis

Table of Contents

[2. Overview 3](#_Toc174699736)

[3. Scope of testing 3](#_Toc174699737)

[4. Types of testing 3](#_Toc174699738)

[4.1. Unit Testing (UT) 3](#_Toc174699739)

[4.2. Component Testing 3](#_Toc174699740)

[4.3. Functional Testing 3](#_Toc174699741)

[4.4. System Testing (ST) 4](#_Toc174699742)

[4.5. End to End Testing 4](#_Toc174699743)

[4.6. Regression Testing 4](#_Toc174699744)

[5. Test Automation Tools and Scenarios 5](#_Toc174699745)

[5.1. Cucumber Framework 5](#_Toc174699746)

[5.2. Selenium Automated Test Framework 5](#_Toc174699747)

[5.3. Open-source Technology stack: 5](#_Toc174699748)

[5.4. Test Automation Test scenarios 6](#_Toc174699749)

[6. Defect Management Approach 7](#_Toc174699750)

[6.1. Defect Discovery 7](#_Toc174699751)

[6.2. Defect Resolution 7](#_Toc174699752)

[6.3. Defects Report for SwagLabs Workshop 7](#_Toc174699753)

[7. Appendix 14](#_Toc174699754)

# Overview

The purpose of this document is to detail testing effort for the SwagLabs service.

# Scope of testing

* SwagLabs website

# Types of testing

The following details the types of functional testing that should be carried out in the delivery of the service.

## Component Testing

Component testing tests a module or component independently to verify its expected output. This is normally performed in the early stages of development so that issues can be found early on. Consider early automation of components, shift left to fail fast approach.

The following components should be tested:

* Login Page
* Products Page (Landing Page)
* Individual Product Pages
* You Cart Page
* Checkout: Your Information Page
* Checkout: Overview
* Finish Page

**Responsibility: Developer/QA**

## Functional Testing

Functional testing verifies whether a deliverable acts in accordance with its pre-determined requirements. Functional testing is only concerned with validating if a system works as intended.

* Boundary value analysis
  + Minimum value.
  + Just above the minimum.
  + Nominal Value.
  + Just below Max value.
  + Max value.
* Field validation
  + Field data type
  + Field length
  + Valid values
  + Mandatory and non-mandatory field entry
* Navigation
  + Buttons navigate correctly
  + Links navigate correctly
  + Search and filter correctly
* Grammar and Spelling
  + Ensure consistency of format across the service
  + Ensure everything is spelt correctly
* Invalid scenarios
  + Mandatory field validation
  + Field length validation
  + Link and button errors
  + Graceful fails – Meaningful error messages

**Responsibility: QA**

## System Testing (ST)

To verify that the delivered functionality correctly fits into the process it is being added to and does not compromise the end to end flow.

* Identify a set of user journeys based on requirements or user stories
* System testing should cover the end-to-end functions of a system

**Responsibility: QA**

## End to End Testing

End-to-End testing tests the entire software product from beginning to end to ensure the application flow behaves as expected. It defines product system dependencies and ensures all integrated pieces work together as expected. Testing should validate:

* Data
* Business and Functional Workflows
* Suitability of the built solution for the business

**Responsibility: QA**

## Regression Testing

To verify that developed or changed components have not compromised the functionality of existing components. Where possible this should be a fully automated test suite and run as part of CI/CD pipeline.

**Responsibility: QA**

# Test Automation Tools and Scenarios

The following outlines the approach and technology stack used for test automation

## Cucumber Framework

Behaviour-Driven Development is methodology that that focuses on defining and verifying the behaviour of applications and services. Cucumber provides an efficient way of automating reusable tests. Some of the benefits of Cucumber are:

* Clarity – the Scenarios are written Gherkin, which uses plain language descriptions.
* Automation – tests can be run continuously and can be run as part of a CI/CD process.
* Reusability – Cucumber scenarios can be used across multiple tests reducing duplication of test scripts
* Traceability – Cucumber scenarios provide traceability across requirements and tests

## Selenium Automated Test Framework

Selenium is an open-source automated testing framework used to validate web application It can be used against most browsers and programming languages. Some of the benefits of using Selenium:

* Languages – You can write scripts in most programming languages and Selenium will convert into Selenium compatible code.
* Open-source – Selenium is publicly accessible automation framework, so no upfront cost.
* Multiple Browser support – Selenium supports most browsers

## Open-source Technology stack:

* Java - openjdk 21.0.4
* Apache Maven 3.6.3
* Selenium 4.23.0
* Junit 5.10
* Cucumber 7.18.1

## Test Automation Test scenarios

The following five scenarios have been written for the SwagLabs website. These test cases correspond to the test cases in the automate test pack.

|  |  |  |  |
| --- | --- | --- | --- |
| tcNo | Acceptance Criteria | Test case title | Test case description |
| SC01 | AC1 - User is be able to login successfully, select a product and checkout | User purchases a single item form SwagLabs website and checks out successfully | * User logs in using valid credential * User navigates to a product and add to shopping cart * User navigates to the shopping cart and checkouts |
| SC02 | AC2 – User is able to login successfully remove a selected item from the shopping cart | User adds and removes an item from the shopping cart on the SwagLabs website | * User logs in using valid credential * User navigates to a product and add to shopping cart * User navigates to the shopping cart and removes item * User logs out |
| SC03 | AC3 – User is able login and access the About page | User accesses the about page | * User logs in using valid credential * User navigates to the About page |
| IV01 | AC04 – User receives a valid errors on the Login page  with invalid credentials | User enters invalid credentials and returns a valid error message | * User attempts to log in with invalid details * User attempts to log in with no credentials |
| IV02 | AC05 – User received valid error on the Complete Order page with invalid customer information | User enters invalid data into the Your Information page | * User logs in using valid credential * User navigates to a product and add to shopping cart * User navigates to the Your Information page and enters no data for each field |

# Defect Management Approach

## Defect Discovery

* Find a defect - Identify defect and reproduce consistently
* Report Defect - Raise the defect and add to the backlog. (Using project management tool such as Jira)
* Triage Defect – Defect needs to be validated and acknowledged as an actual issue.

## Defect Resolution

* Prioritise the risk – Development team analyse and prioritise the defect.
* Fix defect - Defect is fixed based on priority (Backlog refinement to prioritise)
* Update defect with resolution – Defect has been resolved and ready for retest.

## Defects Report for SwagLabs Workshop

Issue 001: User is locked out of the SwagLabs website

**Username:** locked\_out\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter the credentials above and click the login button

**Expected Result:** User should be able to log in and be directed to the Products landing page.

**Actual Result:** User is unable to log in and receives the following error**:**

### *Epic sadface: Sorry, this user has been locked out.*

**Screenshot of error**

A screenshot of a computer

Description automatically generated

Issue 002: User unable to remove item using the remove button

**Username:** problem\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter the credentials above and click the login button

Step 3. From the Products landing page, select a product by clicking on a link

Step 4. Once on the product page click the ‘ADD TO CART’ button. The button text should change from ‘ADD TO CART” to ‘REMOVE’

Step 5. Click the ‘REMOVE’ button

**Expected Result:** Item should be removed from the cart and the button text should return to ‘ADD TO CART’.

**Actual Result:** Button text does not change, remains as ‘REMOVE’ and item remains in basket.

**Screenshot of error**

A screenshot of a computer

Description automatically generated

Issue 003: User unable to add more than two items to the cart

**Username:** problem\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter the credentials above and click the login button

Step 3. From the Products landing page, select two items by clicking the ‘ADD TO CART’ button for each item. The button text should change from ‘ADD TO CART” to ‘REMOVE’ for each item.

Step 5. Try to select a third item.

**Expected Result:** The button text should change from ‘ADD TO CART” to ‘REMOVE’ and the item should be added to the cart.

**Actual Result:** The item text remains as ‘ADD TO CART’ and no item is added to the cart.

**Screenshot of error**

**A screenshot of a computer

Description automatically generated**

Issue 004: User unable to sort items on the Products page

**Username:** problem\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter the credentials above and click the login button

Step 3. From the Products landing page, select the sort droop down at the left hand corner. (For example: Price (low to high)

A screenshot of a computer

Description automatically generated

**Expected Result:** Items on the product page should be sorted by price (low to high)

**Actual Result:** Order of the items does **not** change based on the sort type selected.

**Screenshot of error**

A screenshot of a computer

Description automatically generated

Issue 005: The ‘About page returns 404 not found error

**Username:** standard\_user, problem\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter either of the credentials above and click the login button

Step 3. From the Products landing page, click the menu situated at the top left corner and select the ‘About’ option.

A screenshot of a computer

Description automatically generated

**Expected Result:** the option should navigate you to the SwagLabs About page

**Actual Result:** The option navigates to a 404 Not Found page.

**Screenshot of error**

A screenshot of a computer

Description automatically generated

Issue 006: User encounter latency issues with SwagLabs website

**Username:** performance\_glitch\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter the credentials above and click the login button

**Expected Result:** Products page should load almost instantly

**Actual Result:** There are long delays when logging in and loading the Products page and navigating away from the Products page

Issue 007: The Reset App State does not seem to work

**Username:** performance\_glitch\_user, standard\_user, problem\_user

**Password:** secret\_sauce

**Steps to reproduce:**

Step 1. Open a browser and navigate to the SwagLabs login page (<https://www.saucedemo.com/v1/index.html>)

Step 2. Enter the credentials above and click the login button

Step 4. Make some changes to the Products page. i.e. add an item, sort differently.

Step 3. From the Products landing page, click the menu situated at the top left corner and select the ‘Reset App State’ option.

A screenshot of a computer

Description automatically generated

**Expected Result:** I assume this should reset my changes to the app

**Actual Result:** Does nto appear to do anything at all when selected.

**Screenshot of error**

A screenshot of a computer

Description automatically generated

# Appendix

Draft a brief test plan that includes:

* Types of functional tests you would prioritise (e.g., input validation, boundary testing, user flows).
* A rationale for your chosen testing approach.
* Tools/frameworks you would consider for automation.

Automated Tests:

* Develop up to 5 well-designed automated test cases targeting the website's functionality.
* You’re free to use any technology of your choosing, but please provide the rationale for your selection.
* Include clear documentation for each test case, explaining:
* The specific scenario being tested.
* The reason you chose to automate this particular test.

Issue Reporting:

* If you encounter any bugs or unexpected behaviour:
* Document them in detail (steps to reproduce, expected vs. actual results).
* Explain how you would adjust your testing approach due to these issues.