Test Plan

Introduction

This document outlines the test plan for automating tests on the website https://www.saucedemo.com/ using Playwright with TypeScript. The tests will cover various features, including login scenarios, adding and removing products from the cart, sorting products, the checkout flow, and screenshot comparisons of list items. The automation tests will be stored in a GitHub repository, and GitHub Actions will be used for continuous integration.

Test Item

Web Application: Sauce Demo Website

Objectives

The key Objectives of this project are to;

- Ensure the website's key functionalities are working as expected.
- Automate repetitive tests to increase efficiency.
- Provide a robust and maintainable test suite by leveraging on Page Object Model.
- Integrate automated tests with CI/CD pipeline using GitHub Actions.

Scope

The scope of this test plan includes the following features:

- Login Scenarios (positive/negative)
- Adding and removing products from the cart
- Sorting products in ascending and descending order, price(Low to High) and price(High to Low)
- Checkout flow
- Screenshot comparison of list items

Test Strategy

Tests will be automated using Playwright in a Node.js environment. The tests will be written in TypeScript for better type-checking and maintainability. The project will be structured to follow best practices in organizing test code and managing test data.

Test Environment

- Browser: Chromium, Firefox, WebKit (as needed)
- OS: Windows, macOS, Linux (via GitHub Actions)
- Node.js: Latest stable version
- Playwright: Latest stable version
- GitHub Repository: To store and manage test code
- GitHub Actions: For CI/CD integration

Test Scenarios

Login Scenario (positive/negative)

- Verify that a user can log in with valid credentials.
- Verify that a user cannot log in with invalid credentials (wrong username/password).

Add Product to Cart, Remove from Cart

- Verify that a user can add a product to the cart.
- Verify that a user can remove a product from the cart.

Sorting Products

- Verify that products can be sorted in ascending order.
- Verify that products can be sorted in descending order.
- Verify that products can be sorted by Price High to Low order.
- Verify that products can be sorted by Price Low to High order.

Checkout Flow

- Verify that a user can proceed to checkout
- Verify that a user can input their details before a final checkout
- Verify that the user is shown a price summary by final checkout

Screenshot Comparison of List Items

- Verify that the screenshot of list items is consistent or within acceptable limits by comparing screenshots (base and screenshot to be compared with)

Test Data

Test data will be managed using environment variables for sensitive information (e.g., usernames, passwords, first name, last name, zip code). Other non-sensitive information will be hard-coded or stored in a configuration file.

Test Deliverables

- Test plan
- Test scripts
- Github repository
- GitHub actions

Test Approach

- Manual testing will be used to test all in-scope functionalities of the web application
- Automation testing will be used to test the UI functionality using Playwright with TypeScript

Exit Criteria:

- All the test cases for each scenario must be executed and passed.
- All the test deliverables must be completed and submitted.