**Test Plan for SAUCEDEMO Shop**

Contents

[1. Purpose 2](#_Toc174300188)

[2. Scope 2](#_Toc174300189)

[3. Objectives 2](#_Toc174300190)

[4. Test Items 2](#_Toc174300191)

[5. Features to be tested 2](#_Toc174300192)

[6. Features Not to be Tested 2](#_Toc174300193)

[7. Test Strategy 3](#_Toc174300194)

[8. Scope 3](#_Toc174300195)

[9. Test Approach 3](#_Toc174300196)

[10. Test Prioritization 3](#_Toc174300197)

[11. Testing Types 4](#_Toc174300198)

[12. Test Deliverables 4](#_Toc174300199)

[13. Tools 4](#_Toc174300200)

[14. Test Environment 5](#_Toc174300201)

[15. Entry and Exit Criteria 5](#_Toc174300202)

[16. Testing Resources 6](#_Toc174300203)

[17. Schedule 6](#_Toc174300204)

# Purpose

The purpose of this test plan is to outline the testing strategy for meeting the requirements, software specifications, and design established.

# Scope

Testing will cover the core functionalities including the login process, product list functionalities such as adding and removing items from the cart, checkout process, and some error situations.

# Objectives

* Verify that all functionalities work as expected.
* Ensure the website is user-friendly and meets UI expectations.
* Identify and resolve defects before the store goes live

# Test Items

## Features to be tested

* Login functionality
* Product List
* Add Items to Cart
* Navigation through screens
* Checkout Process

## Features Not to be Tested

* Remove Item from Cart
* Sorting Product List
* Continue Shopping
* Order Completion
* Logout functionality

# Test Strategy

## Scope

The testing will be performed on the product store web application with includes the below functionalities:

* Login
* Logout
* Add/Remove product from cart
* Sorting
* Checkout process
* Order product
* Reset App functionality

## Test Approach

Test case execution will be performed **Automated.**

## Test Prioritization

**Checkout My Information Scenarios**

1. Validate the continue button without filling in any required form fields
2. Validate to fill in only the Firstname and press continue
3. Validate to fill in only the Lastname and press continue
4. Validate to fill only the Zip and press continue
5. Validate to fill only Firstname and Lastname and not zip and press continue
6. Validate to fill only the Lastname and zip and press continue
7. Validate to fill Firstname and Zip and press continue
8. Validate to fill all the form fields and press continue
9. Validate Press cancel

**Checkout Overview Page**

1. Validate product Title, Description and price
2. Validate product quantity
3. Validate the Payment Information
4. Validate the Shipping Information
5. Validate the price total with the product price
6. Validate the total price includes tax
7. Validate Finish button functionality
8. Validate Cancel button functionality

## Testing Types

**Functional testing:** This type of testing ensures that the application functions correctly according to the product store requirements.

**Usability testing:** This type of testing focuses on user experience, ease of use, and user interface design.

## Test Deliverables

* **Test Cases**: Detailed test cases for each feature and functionality.
* **Test Data:** Sample data for product searches, user registrations, and payment transactions.
* **Test Scripts:** Automated test scripts for key functionalities and regression tests.
* **Test Execution Videos:** Video with all test steps recorded in the execution process

## Tools

* **NodeJS**: Is an open-source and cross-platform JavaScript runtime environment. I chose NodeJS (Javascript) because of its high performance and ease of learning. Moreover, NodeJS has a large ecosystem of libraries and packages and supports asynchronous programming which is useful for automation testing.
* **Puppeteer**: Puppeteer is a Node.js library that provides a high-level API to control Chrome. I chose the puppeteer framework because it supports Headless mode which means can run test cases without graphical interface, also provides extensive control over Chrome features such as network conditions, capturing screenshots and generating PDF.
* **Puppeteer-screen-recorder:** This is a Puppeteer plugin that allows you to record videos of the browser's screen while running tests.
* **Mocha:** Mocha is a flexible and widely used JavaScript test framework that runs on Node.js and in the browser. It supports multiple assertion libraries and reporting tools. Mocha excels in handling asynchronous code, including hooks and test organization, allows for writing structured and maintainable test cases.
* **Chai:** Chai is a robust assertion library that integrates with Mocha and other testing frameworks. It provides a range of assertion styles such as expect, should, and assert making it easy to write expressive and readable.
* **Winston:** Winston is a versatile logging library for Node.js that provides extensive customization options for managing log message
* **Chrome For Testing** : This refers to a specific version of the Chrome browser that does not auto-update, unlike the regular Chrome browser. It is particularly useful for testing purposes, as it ensures consistency in the browser environment and behavior across different test runs.

## Test Environment

**Hardware:**

* Laptop/ PC

**Software:**

* Windows OS
* Chrome For Testing Browser
* Test Data
* Test Framework
* Test Cases

# Entry and Exit Criteria

**Entry Criteria:**

* Project Requirements
* Availability of test environment and test data.
* Test Strategy defined
* Code development completed

**Exit Criteria:**

* All critical defects resolved.
* Test coverage meets the specified requirements.
* Execution of all test cases completed

# Testing Resources

* Sokratis Glavinas: Software Test Engineer

# Schedule

* Test Plan Creation: 11/8/2024
* Test Environment Creation: 10/8/2024
* Test Case Development: 09/8/2024
* Initial Test Execution: 11/8/2024

# Improvements

* Performance Testing Metrics
* Validate API requests & responses (API Testing)
* More test coverage