Test Strategy for e-commerce application - SauceDemo

Table of Contents

1. Introduction 3

1.1 Web Application 3

1.2 Scope 3

1.3 Test Objective 3

2 Document Control 3

2.1 Document Author 3

2.2 Document Version History 3

2.3 Review and Approval 3

3 Test Approach 4

3.1 Test Levels 4

3.2 Test Types 4

3.3 Test Automation Framework 4

3.4 Test Environment 4

4 Test Cases 4

4.1 Login Functionality 4

4.2 Product Search 5

4.3 Product filtering 5

4.4 Product Listing and Details 5

4.5 Cart Management 5

4.6 Checkout Process 5

4.7 Logout Functionality 5

5 Test Execution and Reporting 6

5.1 Test Execution 6

5.2 Reporting 6

5.3 Metrics 6

5.4 Defect Management 6

5.5 Maintenance and Updates 6

5.6 Resources and Responsibilities 6

6 Risk and Mitigation 6

6.1 Risks 7

6.2 Mitigations 7

7 Approval 7

7.1 Test Plan Approval 7

7.2 Test Results Approval 7

8 Conclusion 7

1. Introduction

This test plan describes the approach and strategy for validating the important features of an e-commerce application called SauceDemo. The goal is to ensure that the application functions correctly and meets the use requirements. The main functionalities supported b this application are login, logout, product search, filtering, checkout, product listing, product detail and cart management.

1.1 Web Application

- saucedemo.com web application.

1.2 Scope

The below features are the focus area for this automation testing:

- Log In
- Log Out
- Product Filtering
- Product Search
- Cart Management
- Checkout process
- Product listing and detail page

1.3 Test Objective

1. Verify that each feature works as expected
2. Identify, report bugs and fix any functional or usability issues
3. Ensure that the application handles corner cases and invalid inputs gracefully

2 Document Control

2.1 Document Author

- **Name**: Meghana Rao
- **Role**: [Author's Role, e.g., Test Lead, QA Engineer]
- **Contact Information**: mvmeghana@gmail.com

2.2 Document Version History

- **Version 1.0**: [08/09/2024]- Initial version created by Meghana Rao.

2.3 Review and Approval

- **Reviewed by**: [Reviewer's Name, Role, and Date]
- **Approved by**: [Approver's Name, Role, and Date]

3 Test Approach

3.1 Test Levels

- **Unit Testing**: Not applicable (focus on end-to-end testing).
- **Integration Testing**: Validates interactions between components.
- **System Testing**: Ensures the entire application works as a whole.
- **Regression Testing**: Ensures new changes do not break existing functionality.
- **Acceptance Testing**: Confirms the application meets business requirements.

3.2 Test Types

- **Functional Testing**: Validates individual functions and user interactions.
- **Performance Testing**: Ensures the application performs well under load (out of scope)
- **Security Testing**: Assesses application security (out of scope).

3.3 Test Automation Framework

- **Framework**: Pytest
- **Language**: Python
- **Tools**: Selenium WebDriver, pytest-html (for reports)

Choosing the right tools for automation testing is crucial for ensuring efficient and effective test execution. Here's why pytest and Selenium is chosen for this application:

Pytest provides powerful, user-friendly test management and reporting.

Selenium excels at automating web interactions across different browsers. It allows us to create robust, scalable tests and integrates well with other tools for continuous integration and testing

3.4 Test Environment

- **Browser**: Chrome (can be extended to other browsers if required)
- **Operating System**: Windows, macOS, Linux
- **Test Data**: Use predefined data for tests; create mock data if necessary.

4 Test Cases

4.1 Login Functionality

- **Objective:** Ensure users can log in with valid credentials and are prevented from logging in with invalid credentials.
- **Test Cases:**
  Valid login with standard user credentials.
  Invalid login with incorrect username/password.
  Invalid login with locked-out user.

Invalid login with empty credentials.

4.2 Product Search

- **Objective:** Validate search functionalities to ensure users can find products as expected.
- **Test Cases:**
  - Search for a product and verify search results.
  - Check if the search option is present.
  - Check if the search results are accurate and relevant.

4.3 Product filtering

- **Objective:** Validate filter functionalities to ensure users see the products as expected.
- **Test Cases:**
  - Filter using names and expect the right order.
  - Filter using prices and expect the right price order.

4.4 Product Listing and Details

- **Objective:** Ensure product listings and details pages display correct information.
- **Test Cases:**
  - Verify that the product listing page displays all products.
  - Check the product details page for accuracy (price, description, images).

4.5 Cart Management

- **Objective:** Validate cart functionalities including adding, removing items, and updating quantities.
- **Test Cases:**
  - Add items to the cart and verify they appear.
  - Remove items from the cart and verify they are removed.
  - Update item quantities and verify the cart reflects these changes.
  - Ensure cart totals are calculated correctly.
  - Verify empty cart messaging.

4.6 Checkout Process

- **Objective:** Ensure the checkout process is completed successfully and all steps function as expected.
- **Test Cases:**
  - Complete a checkout process with valid information and verify confirmation.
  - Test checkout with an empty cart and verify the appropriate handling.
  - Validate error messages for invalid inputs during checkout.

4.7 Logout Functionality

- **Objective:** Ensure users can log out using the menu option and return to the login page.

- **Test Cases:**
  Successful logout.
  Return to the login page.

.

5 Test Execution and Reporting

5.1 Test Execution

- **Schedule**: Tests will be executed daily as part of the continuous integration pipeline.
- **Execution Method**: Tests will be run using the command-line interface with pytest.

5.2 Reporting

- **Test Reports**: Generate HTML reports using pytest-html for detailed test results.
- **Log Files**: Capture logs during test execution for troubleshooting.

5.3 Metrics

- **Pass Rate**: Percentage of passed tests over total executed tests.
- **Failure Rate**: Percentage of failed tests over total executed tests.
- **Test Coverage**: Measure the percentage of code covered by tests.

5.4 Defect Management

- **Defect Reporting**: Report defects using a tracking system (e.g., JIRA, Bugzilla).
- **Defect Documentation**: Include detailed steps to reproduce, screenshots, and logs.
- **Defect Triage**: Prioritize and assign defects based on severity and impact.

5.5 Maintenance and Updates

- **Test Scripts**: Regularly update test scripts to cover new features or changes in the application.
- **Test Data**: Update test data as the application evolves.
- **Framework Upgrades**: Upgrade testing tools and libraries as needed to keep up with new versions.

5.6 Resources and Responsibilities

- **Test Lead**: Oversees the testing process, ensures adherence to the strategy.
- **Test Engineers**: Develop and execute test cases, report defects.
- **Developers**: Address defects and work with testers to resolve issues.

6 Risk and Mitigation

6.1 Risks

- Application changes affecting test scripts.
- Flaky tests due to network or browser issues.

6.2 Mitigations

- Regularly update and review test scripts.
- Implement retry mechanisms and error handling in scripts.

7 Approval

7.1 Test Plan Approval

- Ensure all stakeholders review and approve the test plan before execution.

7.2 Test Results Approval

- Obtain approval for test results and bug reports from relevant teams.

8 Conclusion

This test strategy document outlines a structured approach to automated testing for the Sauce Demo web application. By following this strategy, we aim to ensure thorough validation of the application's functionalities, maintain high quality, and facilitate continuous improvement.