**TEST PLAN FOR SAUCE DEMO APP**

**Objective**:  
The goal of this test plan is to outline a comprehensive strategy for testing the Sauce Demo application using both **manual testing** and **automation testing** (with **Playwright, Backstopjs and Axe-playwright** as the selected tool). The test plan will cover all key aspects, including functional, UI, visual, and accessibility testing to ensure that the app works as expected across different scenarios.

**1. Test Plan Overview**

* **Application Name**: Sauce Demo
* **Website**: <https://www.saucedemo.com>
* **Purpose**: This is an e-commerce mock application. It includes functionalities like login, product selection, adding items to the cart, and checkout.
* **Testing Types**: Manual and Automation (Playwright)
* **Scope**: Testing all major functionalities, including Login, Product Page, Cart, and Checkout.
* **Tools**: Playwright for automation, Browser DevTools for manual inspection, Backstopjs for Visual automation and Axe-playwright for accessibility automation.
* **Test Environments**:
  + Browsers: Chrome, Firefox, webkit
  + Platforms: Windows

**2. Scope of Testing**

**Manual Testing Scope:**

* Verify the UI elements (buttons, fields, text).
* Validate form functionality (Login, Checkout).
* Perform positive and negative tests.
* Test the user flow from login to checkout.
* Validate error messages and alerts.
* Test compatibility on different browsers (Cross-browser testing).

**Automation Testing Scope**

**-Functional testing (Playwright):**

* Verify the sorting order displayed for Z-A on the “All Items” page.
* Verify the price order (high-low) displayed on the “All Items” page.
* Add multiple items to the card and validate the checkout journey.
* Implement cross-browser testing for Chrome, Firefox, and webkit.

**-Visual testing (Backstopjs)**

* Verify that actual pages match reference screenshots.

**-Accessibility testing (Axe-Playwright)**

* Verify that pages don’t have accessibility voilations .

**Features to be Tested:**

1. **Login Functionality**:
   * Valid and invalid credentials.
   * Locked-out user behavior.
2. **Product/Inventory Page**:
   * Display of products and their details.
   * Sorting and filtering options.
   * Adding/removing items from the cart.
3. **Cart Functionality**:
   * Correct items and quantities displayed.
   * Price calculation.
4. **Checkout Process**:
   * Input validation for shipping details.
   * Completion of orders.
   * Verifying final checkout and order confirmation.
5. **Logout Functionality**

* Logout successfully

1. **UI/UX**:
   * Layout consistency across pages.
   * Responsive design.

**3. Test Objectives**

* **Functional Testing**: Ensure all functional aspects of the Sauce Demo app work as intended, covering all core user journeys.
* **Cross-Browser Testing**: Verify that the app works consistently on major web browsers.
* **Usability Testing**: Ensure ease of use, with an intuitive interface and smooth navigation between pages.
* **Accessibility Testing**: Ensure app can be used by users with disabilities, including those with visual, auditory, motor, or cognitive impairments. And ensure the app adheres to accessibility standards.
* **Visual Testing**: Ensure pages are aligned with design reference

**4. Testing Strategy**

**Manual Testing Strategy**

1. **Exploratory Testing**:
   * Navigate through the app without predefined test cases to discover potential issues in navigation, functionality, and layout.
2. **Test Case Execution**:
   * Execute predefined test cases for each feature (Login, Product Page, Cart, Checkout).
3. **Cross-browser Testing**:
   * Manually verify the behavior on Chrome, Firefox, and Safari.
4. **Form Validation**:
   * Enter valid and invalid data in the login and checkout forms and verify the app's responses.
5. **Edge Case Testing**:
   * Test the application with edge case inputs like invalid credentials, special characters, and maximum field lengths.

**Automation Testing Strategy (Playwright, BackstopJS, Axe-Playwright)**

**1. Functional Testing (Playwright)**

**1.1. Setup Playwright:**

* Install Playwright and configure test scripts to run across Chrome, Firefox, and WebKit browsers.

**1.2. Test Script Creation:**

* **Product Tests:**
  + Verify the sorting order (Z-A) on the "All Items" page.
  + Verify the price order (high to low) displayed on the "All Items" page.
* **Checkout Test:** Add multiple items to the cart and validate the entire checkout journey, including:

**1.3. Cross-Browser Testing Automation:**

- Execute the same set of tests across multiple browsers to ensure cross-browser compatibility.

**1.4. Parallel Execution:**

- Configure Playwright to run tests in parallel for faster feedback.

**1.5. Bug Logging:**

- Automatic Logging: Capture screenshots and logs for failed test cases to assist in debugging.

**2. Visual Testing (BackstopJS)**

**2.1. Setup BackstopJS:**

* Install BackstopJS to automate visual regression testing by comparing UI screenshots against a baseline.

**2.2. Test Scenarios:**

* **Login Page Visual Test**: Capture a baseline image of the Login page.
* **Product Page Visual Test**: Capture a baseline image of the product listing page, focusing on the correct display of product sorting and prices.
* **Cart and Checkout Visual Test:** Validate the visual consistency of the cart and checkout pages during the user journey.

**2.3. Execution:**

* Run BackstopJS visual tests across different browsers and viewport sizes to ensure consistent UI rendering.
* If differences are detected, BackstopJS highlights the differences between the baseline and current images.

**2.4. Bug Logging for Visual Tests:**

- Automatically generates a visual comparison report showing pixel differences.

**2.5. Visual Test Workflow:**

1. Set up baseline images for all key UI components.

2. Run visual tests during every build and compare screenshots against baselines.

3. Log any differences in the test report.

**3. Accessibility Testing (Axe-Playwright)**

**3.1. Setup Axe-Playwright:**

* Install Axe-Playwright to perform accessibility checks based on WCAG 2.1 standards.

**3.2. Test Scenarios:**

* **Login Page Accessibility Test**: Ensure that the username and password fields, as well as the login button, are accessible with proper labels, error messages, and focus management for screen readers and keyboard navigation.
* **Product Page Accessibility Test**: Ensure product names, images, and buttons have accessible labels and are keyboard-navigable.
* **Cart Page Accessibility Test**: Ensure the cart items, quantities, and checkout buttons are labeled for screen readers, and all actions are keyboard-navigable.
* **Checkout Step One Page Accessibility Test**: Ensure the form fields for user information (first name, last name, postal code) have accessible labels, and focus is properly managed when navigating between fields.
* **Checkout Page Accessibility Test**: Ensure all form fields are accessible, with proper labels and focus handling.

**3.3. Execution:**

* Integrate accessibility checks within Playwright tests using Axe-Playwright.
* Run accessibility tests across different pages to ensure compliance with WCAG.

**3.4. Bug Logging for Accessibility Tests:**

* Accessibility violations are logged, with detailed suggestions on fixing them.

**3.5. Accessibility Test Workflow:**

1. Run accessibility tests alongside functional tests to verify compliance.

2. Log accessibility violations in the report with suggested fixes.

**5. Test Environment Setup**

**Manual Testing Environment:**

* Browsers: Chrome, Firefox, Safari
* OS: Windows 10

**Automation Testing Environment:**

* Install Node.js, Playwright, BackstopJs and Axe- playwright.
* Ensure browser drivers for Chrome, Firefox, and webkit are up to date.

**6. Test Cases**

**Manual Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Title** | **Steps** | **Expected Result** |
| TC01 | Login with valid user | 1. Go to login page: <https://www.saucedemo.com/>  2. Enter valid credentials.  3. Click login. | User is logged in and redirected to the Inventory/Products page. |
| TC02 | Login with invalid user | 1. Go to login page.  2. Enter invalid credentials.  3. Click login. | Error message displayed: “Username and password do not match any user in this service.” |
| TC03 | Login without any user credentials | 1. Go to login page.  2. Click login. | Error message displayed: “Username is required.” |
| TC04 | Login with only username credentials | 1. Go to login page.  2. Enter only username credentials.  3. Click login. | Error message displayed: “Password is required.” |
| TC05 | Login with only password credentials | 1. Go to login page.  2. Enter only password credentials.  3. Click login. | Error message displayed: “Username is required.” |
| TC06 | Change sorting to Z-A | 1.Log in.  2.Select dropdown option “Name(Z to A) | Products are sorted in descending alphabetical order (Z-A). |
| TC07 | Change price sorting (High-Low) | 1.Log in.  2.Select dropdown option “Price(High to Low) | Products are sorted by price from highest to lowest. |
| TC08 | Change price sorting (Low-High) | 1.Log in.  2.Select dropdown option “Price(Low to High) | Products are sorted by price from lowest to highest. |
| TC09 | Add product to cart | 1. Log in.  2. Go to the Products page.  3. Click on a product's "Add to cart" button. | Product is added to the cart, Cart icon updates. |
| TC10 | Remove product from cart | 1. Log in.  2. Go to the Products page.  3. Click on a product's "Add to cart" button.  4. Click on cart icon to go to Cart age  5. Click on a product’s “Remove” button | Product is removed from the cart |
| TC11 | Complete checkout step-one process | 1. Add items to the cart  2. Go to the cart page.  3. Click “Checkout” button to go to Checkout-step-one page  5. Fill “First Name”, “Last Name” and “zip/postal code” text fields  6. Click “Continue” button | Checkout step one process is completed and user is navigated to Checkout step-two page |
| TC12 | Complete checkout | 1. Add items to the cart.  2. Go to the cart and click Checkout.  3. Fill in shipping info and click Continue.  4. Click Finish to complete checkout | Order is completed, and order confirmation is displayed. |

**Automated Test Cases**

* 1. **Functional Test cases (Playwright)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Title** | **Script Description** | **Expected Result** |
| FTC01 | Login Test | Automate login with valid credentials and check redirection. | User is logged in successfully. |
| FTC02 | Sorting by Z-A | Automate and verify the sorting order displayed from Z-A on the "All Items" page. | Products are sorted in descending alphabetical order (Z-A). |
| FTC03 | Price Sorting (High-Low) | Automate and verify the price sorting order (high to low) on the "All Items" page. | Products are sorted by price from highest to lowest. |
| FTC04 | Add Multiple Items & Checkout | Automate the process of adding multiple items to the cart and validate the entire checkout journey. | Multiple items added, cart total updated, successful order confirmation. |

* 1. **Visual Test cases (BackstopJs)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Title** | **Script Description** | **Expected Result** |
| VTC01 | Login Page | Automate snapshot reference and test snapshot of login page. | Reference and test snapshots match. |
| VTC02 | Inventory Page | Automate snapshot reference and test snapshot of Inventory page. | Reference and test snapshots match. |
| VTC03 | Cart Page | Automate snapshot reference and test snapshot of Cart page. | Reference and test snapshots match. |
| VTC04 | Checkout step-one page | Automate snapshot reference and test snapshot of Checkout step-one page. | Reference and test snapshots match. |
| VTC05 | Checkout step-two (Overview) page | Automate snapshot reference and test snapshot of Checkout step-two page. | Reference and test snapshots match. |

* 1. **Accessibility Test cases (Axe-Core)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Title** | **Script Description** | **Expected Result** |
| ATC01 | Login Page | Automate and verify accessibility violations of login page. | No accessibility violations found |
| ATC02 | Inventory Page | Automate and verify accessibility violations of Inventory page. | No accessibility violations found |
| ATC03 | Cart Page | Automate and verify accessibility violations of Cart page. | No accessibility violations found |
| ATC04 | Checkout Page | Automate and verify accessibility violations of Checkout page. | No accessibility violations found |

**7. Exit Criteria**

* All critical and high-severity bugs must be resolved.
* 100% execution of test cases, with all functional tests passing.
* No blocker or critical bugs open at the time of product release.
* Test coverage for at least 90% of the features.

**8. Reporting**

* **Manual Test Reporting**:
  + Maintain an Excel/Google Sheets document for all executed test cases, capturing Pass/Fail status.
  + For failed cases, provide details of the issue, along with screenshots.
* **Automation Test Reporting**:
  + Configure Playwright to generate test reports automatically.
  + Integrate a Visual testing tool (BackstopJs)
  + Integrate a Accessibility testing tool (Axe- playwright)

**9. Risk Management**

* **Automation Script Failures**: To mitigate risks of automation failures, regularly update the Playwright scripts to reflect UI changes and maintain stable test data.
* **Cross-Browser Compatibility**: Test the app on different to ensure consistency of app key functionalities across popular browsers
* **Updating Script**: Track updates, changes and work with cross-functional team by using version control (Github)
* **Maintainability of Script:** Implement modular framework by using page object design pattern to improve maintainability, scalability, and readability of scripts over time.

#### 10. ****Conclusion****

The plan outlines a comprehensive strategy for testing the Sauce Demo App, integrating manual and automated testing with Playwright, BackstopJs and Axe- playwright. The manual tests will cover exploratory and detailed functional testing, while Playwright will handle cross-browser, and functional tests of assessment scope. This approach ensures a robust and efficient testing process that adheres to QA best practices.