Automation Test Plan for Sauce Demo Project

This plan should is used by automation developers to provide an automation plan which can be attached to the Test Plan. The sections in this document allow the developer to share the required information with the team and management as necessary. Not all sections will be requested or known depending on individual projects.

Document History

Version	Date	Author	Description
1.0	5 Dec 2023	Pankaj shinde	First draft for review

Sign-off or Acknowledge

To sign off or acknowledge this test plan, click **here** or the image above.

Table of Contents

Summary and Introduction

The majority of the tests that will be created for this project will be required for on going regression testing. To save time and resources going forward, these tests will be automated. Test automation will also enable more streamlined approach to creating the tests and provide a repeatable level of validation. We need to thoroughly test the functionality, usability, and performance of the https://www.saucedemo.com/inventory.html web application.

Automation Test Scenario Selection Criteria

The automate-able scenarios will be selected only after the approval of all stake holders in the team during the Sprint planning session.

The scope of automate-able test scenario is decided during spring planning of each story. The automate-able test scenario selection criteria are given below, but not limited to

- Scenarios which verify data exchanges between systems such as Queues, Adapters and Databases
- Scenarios which focuses on system and integration against end to end
- Scenarios that focuses on API layers against GUI level.
- Scenarios which are data driven which causes more time to execute manually.

Other reasons for not automating a scenario can be checked here.

Scope and coverage

This section is a place holder to add the in scope or out of scope systems/components. There should be mentioning about whether the automation is completely new or an extension/modification of an existing automation.
Automation will be used for verifying the systems for scenarios which are only specific to this project. Automation will cover the following components but may not be limited to:

Feature According to Module:

Name	**In Scope**	**Coverage**
Functional Testing	• Validate login functionality. • Verify product listing and sorting. • Test cart functionality. • Ensure checkout process works correctly.	100%
Compatibility Testing	• Test across different browsers (Chrome, Firefox, Safari, Edge). • Validate responsiveness on various devices (desktop, tablet, mobile).	100%
Performance Testing	• Evaluate load time of the website. • Check responsiveness under varying network conditions.	100%
Security Testing	• Ensure the application is secure against common vulnerabilities (SQL injection, XSS, etc.).	100%
Data Validation	• Ensure the accuracy of data returned in responses. • Validate that product details match with the front-end application.	100%

Databases*:*

Name	**In Scope**	**Coverage**
User Access	Verification will be done on tables which holds data related to Login, Enter Checkout detaills, ETF Composition data.	100%
Orders placed information	Verification will be done on tables which holds data related to Checkout details and account details.	100%

Webservices:

Name	**In Scope**	**Coverage**

Authentication	• **Endpoint:** `/login` • **Methods:** POST • **Payload:** JSON with username and password • **Expected Response:** JSON response with authentication token or error message	100%
Product Retrieval	• **Endpoint:** `/products` • **Methods:** GET • **Expected Response:** JSON array containing product details	100%
Product Details	• **Endpoint:** `/products/{product_id}` • **Methods:** GET • **Expected Response:** JSON object with specific product details	100%
Cart Management	• **Endpoint:** `/cart` • **Methods:** GET, POST, DELETE • **Payload (for POST):** JSON with product details • **Expected Response:** JSON response confirming product addition/removal or cart details	100%
Checkout Process	• **Endpoint:** `/checkout` • **Methods:** POST • **Payload:** JSON with checkout details (address, payment information, etc.) • **Expected Response:** JSON response confirming successful checkout or error message	100%

GUI:

Name	**In Scope**	**Coverage**
Authentication	• Input fields for username and password. • Buttons to trigger login and view the result. • Display area to show the response (success or error message).	100%
Product Retrieval	• Button to fetch product details. • Display area to present the retrieved product details.	100%
Product Details	• Input field to enter a product ID. • Button to retrieve details. • Display area to show the specific product's details.	100%
Cart Management	• Buttons to add, remove, and view cart contents. • Display area to show the response for cart actions.	100%
Checkout Process	• Input fields for checkout details (address, payment info). • Button to trigger checkout. • Display area to show the success or failure of the checkout process.	100%

Out Of Scope

1. **Not within Test Objectives:** The activities, features, or functionalities associated with the inventory page of "https://www.saucedemo.com/inventory.html" are not within the scope of the current testing phase or project objectives.
2. **Not Allocated Resources:** The team might not have allocated resources, time, or expertise to cover testing or analysis for that particular URL.
3. **Specific Testing Exclusions:** There could be a deliberate decision to exclude testing or analysis for this page due to specific reasons like time constraints, lower priority, or it might be covered in a different phase of testing.

Standards & Processes

Automation test scripts will be developed in accordance to SauseLabs's test automation standards & guidelines. To summarise, the development of the automation script is done in the automation testers' desktop and it is unit tested locally. The code is checked in to GIT into a branch specific for that feature. Before merging to the master, the code is given for the peer review by; either member of the test automation team or qualified Project team members.

Validation

For each test case, all messages being sent and received will be validate against the control data.

Features:

- User-friendly input fields and buttons for each test category.
- Clear and readable display areas to show the API responses or relevant information.
- Responsive design for different screen sizes.
- Proper error handling to display error messages or alerts.

Functionality:

- Validate user inputs (e.g., valid username/password, correct product ID).
- Trigger API calls corresponding to the test categories.
- Display API responses or any error messages in the designated display areas.
- Allow users to interact with the test plan functionalities seamlessly.

Automation Approach

The scenarios will be system tested or integration tested using the automation tools. Pre-selected or controlled data will be used to inject into the system in the form of XML, Webservices, CSVs, Records in databases and the expected output will be verified in Databases or output XMLs. Data driven testing will be used to cover scenarios such as boundary values, class types and exception cases. The existing SauseLabs codes in the system will be used and modified to cover the tests. Initially the automation scenarios cover the system testing cases and as Sprints are in progress, new integrated scenarios will be added by extending the data injection or validation to outer connected systems.

Tool Set / Framework

Since the test scenarios are in the format of feature files, cucumber with java is selected as the automation execution tool framework. Visual Code Nugget packages will be used for the UI automation part. The code is versioned using Git and stored in Stash.

- Use a programming language with a GUI library or framework (Playwright, BDD, Cucumber, Gherkin, TypeScript).
- Integrate APIs and backend logic corresponding to each test category.
- Ensure a user-friendly and intuitive design.

Environment

Automation will be executed in DEV environment during each Sprint. Regression Automation test will be done on TB3 Environment. No automation will be executed on E2E environment.

- Test on multiple browsers: Chrome, Firefox, Safari, Edge.
- Use different devices: Desktop, Tablet, Mobile.
- Utilize tools like Selenium WebDriver, and BrowserStack for testing.
- Use tools like Postman, RestAssured, or similar libraries/frameworks for API testing.
- Maintain separate test environments for development, staging, and production.

Timelines

Test Automation will be developed for the selected scenarios during the same sprint and will be delivered in the next sprint. A regression pack will be chosen from this set to be run during Regression Testing Phase.

Reporting and Dashboard

Once the tests are executed, it's imperative to understand if there were any failures and why those failures occur in order to make the framework more robust. Document test results, including screenshots and logs.

- Provide detailed reports for each test scenario.
- Maintain traceability matrix mapping test cases to requirements.
- Document test cases, including endpoint details, payloads, and expected responses.
- Record test results, including any issues encountered and their severity.
- Generate reports with detailed test logs and screenshots if necessary.

Regression And Maintenance plans

(i) Regression Plan:

New Scripts and Regression Library:

- **New Scripts Inclusion:** Yes, the new scripts for testing https://www.saucedemo.com/inventory.html will be added to the regression library.
- **Scheduled or Adhoc Basis:** They will be run as part of the regression suite on a scheduled basis, ideally after each build or deployment.

Responsibilities:

- **Monitoring & Investigation:** The QA team or designated testers will be responsible for monitoring regression results.
- **Maintaining & Escalating:** They will investigate any failures, maintain the scripts, and escalate critical issues to the development team.

Regression Environments:

- **Ongoing Regression Environments:** Multiple environments will be used for ongoing regression, including staging, pre-production, and production-like environments to mimic real-world scenarios.

(ii) Maintenance Plan:

Impact on Existing Test Cases:

- **Maintenance Requirement:** Maintenance might be required due to changes in the application. Regular updates to test scripts or cases are anticipated.
- **Updated Test Cases:** Existing test cases related to inventory management, authentication, product retrieval, cart management, and checkout processes will need updates to adapt to changes in the implemented software.

Reasons for Maintenance:

- **Software Implementation Impact:** The software implementation might introduce changes in the UI, functionality, or backend logic, necessitating updates to existing test cases to ensure continued compatibility and validity.

Assumptions, Dependencies, Risks and Mitigation

- High level test scenarios will be ready and approved by BA, Developer and Testers before the automation development starts for a feature
- All necessary accesses to the Databases, Application are available to the automation developer
- Feature development is completed before the automation test code is executed.
- Changes in the requirement may impact the existing automation test cases and additional effort need to be provided after review.

Sample Risks / Mitigations:

Description	Likelihood	Impact	Mitigation	Owner
Testing API to be provided by developers for Application baseline testing required for automated testing get deprioritised.	Low	Medium	Changes in test approach e.g. test data to be fed to source system database and verification to be done against downstream application.	Test Manager
Firewall and connectivity required for automation are not available	Low	High	Certain test scenarios will require to be covered by manual testing	Test Manager

Assumptions / Dependencies :

<Risks / Mitigation>

- Identify potential risks like compatibility issues or performance bottlenecks.
- Develop mitigation strategies to address identified risks.