**TEST PLAN**

Project Name: Swag Labs Testing

Test Engineer: Basil

Date: 24/12/23

Prepared by: Basil

Reviewed by: Srirangam Bhavani

**1. INTRODUCTION**

This test plan outlines the testing strategy for the Swag Labs application using the TestNG testing framework with the page object model.

**2. Aim/Objective**

- Validate the functionality of Swag Labs.

- Identify and report defects or issues in the tested functionalities.

- Ensure that Swag Labs meets the specified requirements.

**3.Scope of testing:**

|  |  |  |
| --- | --- | --- |
| * Module Name | Applicable Roles | Description |
| Account Log in | USER | Users can login using the username and password |
| Filtering | USER | Customer Can filter items using the dropdown menu |
| Add-to-Cart& Checkout | USER | Customers can merchandise to cart which they like and proceed to checkout |

* + Within the scope:
  + Functional Testing,
  + External interface,
  + Browser compatibility.
  + Automation Testing
  + Out of scope:
    - Non-functional testing, (stress, performance)

**4)Test Strategy:**

1. Levels of Testing:

* **Unit Level**: Test individual code components in isolation to ensure their correctness, helping catching defects
* **Integration with System**: in this testing we test the end-to-end features. In this testing we are going to test whether all functions of creating account, account login, searching for pets and Buying pets are working as per the client requirements or not.
* **UAT (User Acceptance Test):** In this testing we test whether the application can handle real time scenarios are not.

1. Types of Testing:

* **Functional testing**: It is the technique that helps to verify the functionality of the software by the given modules like **Account Registration**, **Account Sign-in**, **searching pets**, **Add-to-Cart& Checkout**.
* **Smoke testing**: It is the technique that we can test basic and critical bugs in application before going to the one round of regression testing.
* **Sanity testing**: In this testing we are going to check that all builds of the software are fixed are not.

**5.Entry and Exit Criteria:**

1. Entry Criteria:

* Requirements should be well-defined and approved.
* Sufficient test data should be available.
* Test environments must be set up.
* Test cases should be prepared and reviewed.

1. Exit Criteria:

* All critical functionalities are tested and meet acceptance criteria.
* Bugs that are remaining must be fixed.
* Test scripts must be executed 99%.
* Every critical bug must be removed.
* All high-priority defects are addressed.

**6.Test Deliverables:**

* Test plan document
* Test case document
* Test tools
* Test data
* Test results and reports
* Defect reports
* RTM

**7.Risk and Mitigation:**

**Incomplete Test Coverage:**

•Risk: Not covering all aspects of the application in the test suite.

•Mitigation: Regularly review and update the test suite to include new features. Use code coverage tools to identify areas not covered by tests.

**Data Dependency:**

•Risk: Tests relying on specific data, such as login credentials, may fail if the data changes.

•Mitigation: Maintain a stable and controlled test environment. Use test data generation tools to ensure the availability of required data.

**Environmental Issues:**

•Risk: Test failures due to inconsistencies or differences between test and production environments.

•Mitigation: Ensure that the test environment closely mirrors the production environment. Regularly synchronize test data and configurations.

**Browser Compatibility:**

•Risk: Tests may pass in one browser but fail in another.

•Mitigation: Include cross-browser testing in the test plan. Test on popular browsers to ensure compatibility.

**Test Data Quality:**

•Risk: Poor quality of test data may lead to inaccurate test results.

•Mitigation: Validate and sanitize test data. Use realistic and diverse data to simulate various user scenarios.

**Test Execution Dependencies:**

•Risk: Some tests may depend on the successful execution of others, leading to a cascading effect if one fails.

•Mitigation: Clearly define test dependencies. Execute tests independently where possible. Use test automation frameworks that support parallel execution.

**Code Changes During Testing:**

•Risk: Changes to the application code during the testing phase may affect test results.

•Mitigation: Freeze the codebase during critical testing phases. Communicate code freezes to development teams.

**Test Script Maintenance:**

•Risk: Difficulty in maintaining and updating test scripts as the application evolves.

•Mitigation: Implement a robust test automation framework. Regularly review and refactor test scripts. Provide training for test script maintainers.

**False Positives/Negatives:**

•Risk: Tests producing inaccurate results due to false positives (incorrectly passing) or false negatives (incorrectly failing).

•Mitigation: Regularly review and update test assertions. Investigate and address any inconsistent test results.

**8.Test environment:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Test tool/Environment** | **Purpose Details** |
| 1 | Browser | * Chrome (Latest version) * Firefox (Latest version) * Safari (Latest version) |
| 2 | Operating system | * Windows 10 * MacOs * Linux (Latest version) |
| 3 | Network connection | * Stable and uninterrupted network connectivity to access the application, share files, and collaborate effectively. |
| 4 | Test tools | * Various testing tools and frameworks required for automation (e.g., Selenium WebDriver for web automation, JUnit/ TestNG for test execution, etc.). |

**9. Assumptions:**

O The System is assumed to be easily manageable and navigable by end-users.

O The Software is assumed to be free of critical errors or major defects.

O Test case design activities will be undertaken by the QA group.

O The necessary tools and hardware have been allocated and are ready for use in the testing process.

**10.Approval Information:**

This project needs to be approved by project manager

**Signature:**

* Name: Srirangam Bhavani
* Role: Project manager
* Date: 26-01-2024

**11.Test metrics:**

* Test case pass percentage-(No. of test cases passed/Total No. of test cases) \* 100.
* Test case failed percentage-(No. of test cases failed/Total No. of test cases) \* 100.
* Defects deferred percentage-(No. of defects deferred/Total No. of defects) \* 100.
* Test Cycle Time: Measures the time it takes to complete a test cycle or testing phase.