Contents

[Assignment – SwagLabs 1](#_Toc181364625)

[Test automation code (selenium/Java/TestNG) 1](#_Toc181364626)

[Other Test Documents 2](#_Toc181364627)

[Test cases (also refer Testplan.xlsx) 2](#_Toc181364628)

[Test Approach (Please also see MasterTestPlan\_Swaglabs.docx for creating test plan format) 3](#_Toc181364629)

[Test Data 3](#_Toc181364630)

[Test Automation/CICD delivery 3](#_Toc181364631)

[Why To automate 3](#_Toc181364632)

[Test management tools 3](#_Toc181364633)

[Defect management 4](#_Toc181364634)

[Defects Reported 4](#_Toc181364635)

# **Assignment – SwagLabs**

All the documents are committed to TestDocuments folder in GIT repository. Please see the path

below

<https://github.com/SarilTest/Swag_Repo>

# **Test automation code (selenium/Java/TestNG)**

Test can be executed running the testNG class files or as testNG suite which execute 5 test classes together (for major features) and have total of 18 tests in total. 2 of those tests will show as failed and it’s intended to fail as per the use case planned.

<https://github.com/SarilTest/Swag_Repo/tree/main/src/test/java/project>

Latest execution results after final commit.

===============================================

Suite

Total tests run: 18, Passes: 16, Failures: 2, Skips: 0

===============================================

Details of the failure tests.

1. Performance glitch user – As this is expected to be a normal user login in real time, I have created test to fail this using the performance Glitch user to show that this will fail during test execution and can raise as a defect for fixing.
2. Boundary value for “your Information- POSTAL CODE” validation. This test case should not accept more than 10 characters in the postcode, so I have added a test with 11 characters and this test expected to be failing

# **Other Test Documents**

<https://github.com/SarilTest/Swag_Repo/tree/main/TestDocuments>

Test Document info

1. Testplan.xlsx – describes few major user stories, use cases, test cases (manual)
2. MasterTestPlan\_Swaglabs.docx - Draft version of overall test plan for the SWAGLABS testing project- This is a draft version describing the overall testing approach, and other test plan contents. I have only filled the relevant or critical areas to describe how this ca be done.
3. BugReport311024- Bugs reported, also created manual excel bug reporting format for 3 issues. Also added a screenshot of JIRA bug report as it will be more convenient to use and track. (Not all the bugs are reported, also not framed)
4. AssginentOverallInfo.docx. (this documents) details all activities I have performed for this assignment.

# **Test cases (also refer Testplan.xlsx)**

<https://github.com/SarilTest/Swag_Repo/tree/main/TestDocuments>

The critical test cases are available in the below document, I have created few user stories and drafted manual test cases for some of the major functionalities. Please see more detailed information in Testplan.xlsx document for detailed test cases.

The functional tests I found more critical to test I would consider as per the navigation and major user flows are below

1. Verify that valid user can login and invalid logged-in user can’t login
2. Verify login with valid/invalid username and password using boundary value considerations
3. Verify that a Valid logged-in user can see a list of all available products.
4. Verify that a logged-in user can see product details (name, price, and image).
5. Verify that a logged-in user can sort products by price and name.
6. Verify that a user can view the detailed page of a product clicking individual product from catalogue
7. Verify that a user can add products to their cart from both the product catalogue and product details page.
8. Verify that a user can Complete the check-out process after validating the cart and shipping details from the confirmation page
9. Verify that the website loads quickly and pages transition smoothly without any performance issues.
10. Verify that the website redirects to the login page when a logged-in user state URL is copied and pasted into another browser or window.

# **Test Approach (Please also see MasterTestPlan\_Swaglabs.docx for creating test plan format)**

[**https://github.com/SarilTest/Swag\_Repo/tree/main/TestDocuments**](https://github.com/SarilTest/Swag_Repo/tree/main/TestDocuments)

Test approach is briefly described defined in the Master test plan document which that includes various types of testing techniques to cover different aspects of the functionality, performance, security, and usability of the application. Testing should consider validating the major test requirements like login page, security, Login to check out user flows, input field validations, boundary value validations, compatibility with browsers, performance and overall seamless user experience throughout browsing.

## **Test Data**

Test approach consideration for test data creation can be done manually or can consider test data creation tools like Data faker or MS excel Power Query etc.

## **Test Automation/CICD delivery**

Test automation considers using Selenium JAVA and TestNG as this is free and most stable testing framework at the moment in the market and widely used. Also, this supports well with dev approach considered for the project to locate the elements for cross browser testing. The selenium WebDriver directly interacts with web elements and same manual test cases can be converted with ease. Even though a company dedicated support is not possible, we should take the advantage of large community support is available over the web and selenium website for any type of issues facing up to date. Testers with a reasonable experience on automation can easily adapt to this framework. Selenium TestNG integration with CICD tools and framework is very much possible and can be done without much dedicated support. There are lot of plugins available for additional reporting and streamlining the process. TestNG is very successful in arranging and organizing the tests and can manage the individual and Parallel execution of cross browser testing. Also test prioritization is possible using TestNG annotations, parameterization and dependency also can be managed using variables and data input. Selenium JAVA, TestNG is a well-known automation choice for many due to its simplicity, support, popularity among developers, industry standards etc.

## **Why To automate**

Many repeated use cases related to Login, Boundary values field limits, User inputs can take a lot of manual stress and time to execute. Negative testing in many user flows can have automated suite all the time for regression testing. E2E scenarios can be easily run multiple times for each major release and regression suite execution. Automation gives correct error messages all the time every time of its execution, manually this is very difficult to replicate each time. So, the expectation can be predicted and its error free. All automation suites can be reused in future or similar projects and also for all BAU releases. If using TESTNG you can also organise the test suites in different levels of execution using annotations.

## **Test management tools**

JIRA and Confluence can be considered for test management. JIRA can be connected with wide range of automation tools and frameworks; Plugins are available from test NG reporting.

If the project follows **AGILE** way of working, JIRA helps well in managing the requirement, test cases, and test reporting with various planning (Sprint plan, Kanban boards, Dashboards, Xray integration for automation test suite execution) and reporting. If following SPRINTs, we can easily have all the Epic, feature, backlogs added in project and can have sprints panned using product /sprint backlogs. We can have various breakdown charts, two dimensional dashboards in JIRA management.

## **Defect management**

For Defect management, we can use the same JIRA and JIRA bug reporting structure is very comprehensive and organised. JIRA defects can be mapped to use cases, test cases and other requirements to create traceability. Anyone can easily learn and adapt to JIRA and Confluence for document generation and support.

## **Defects Reported**

<https://github.com/SarilTest/Swag_Repo/tree/main/TestDocuments>

Defects found while exploratory testing

1. Username and password boundary validation fails in login
2. Field validation and boundary value checks fails in checkout - your information input fields
3. URL state is not reset to login page when copy page in another window (able to continue without login)
4. Not able to edit the item quantity from the cart.
5. Can go back to logged in stage even after logout using back button (Security)
6. No option to add or remove product from the checkout overview page
7. User able to checkout even after removing all cart items from the cart
8. No payment page is displayed during checkout process, able to finish the order without payment (might be expected in this website as the integration is not done yet)
9. The add to cart button is not changed to REMOVE button when product is added from the single item description page, Also the Cart badge does not show that item added.
10. User logged-in Info(account) is not shown anywhere after logged in
11. The footer links are not enabled.
12. The image(logo) near to the products header in product detail page (all items) is not properly aligned

Testing needs to be carried out in manual test execution until the code is stabilised and user flow defect are minimised. Test plan Entry and Exit criteria defined how this can be achieved and move to the regression to consider automation for stable features. Shift left or risk-based testing can suggest and can be initiated based on the stable areas and functional and regression tests can be automated accordingly. For e.g. The login module is more stable and can be automated once the boundary value tests are fixed and resolved.