**Documents I created/used for testing purposes :**

1. **Bug report**
2. **Test Plan**
3. **Test case**
4. **Test scenario for automation testing with Selenium**
5. **Test Summary Report**
6. **Calendar test with Selenium - code**

**Bug Report**

**Bug ID:** WB-001  
**Reported by:**   
**Date Reported:**

**Bug Summary**

**Title:** Dropdown menu fails to display options on the settings page.  
**Description:** On the settings page, the "Language" dropdown menu does not display any options when clicked, preventing users from changing their preferred language. This issue occurs across multiple browsers.

**Steps to Reproduce**

1. Navigate to the application's settings page.
2. Scroll to the "Language" dropdown menu.
3. Click on the dropdown arrow.

**Expected Behavior**

The dropdown menu should display a list of available languages for selection.

**Observed Behavior**

Clicking the dropdown arrow does not show any options, and no action occurs.

**Environment**

**Operating System:** Windows 10  
**Browser:** Chrome (Version 116.0), Firefox (Version 118.0)  
**Device:** Desktop  
**Application Version:** 2.4.1

**Severity**

High - Users are unable to modify language preferences, potentially affecting accessibility for non-English speakers.

**Attachments**

* [Screenshots or video of the issue]
* [Browser console logs]
* [Network logs, if applicable]

**Additional Information**

The issue was also replicated in incognito mode to rule out browser extensions.

**Test Plan**

**1. Introduction**

The main goal of testing activities is to provide information about the quality of the product under test.

**2. Scope of Testing**

Types of tests to be conducted:

* Unit tests
* Functional tests
* Performance tests

Types of tests that will not be conducted:

* Automated tests – due to insufficient budget for the software automation testing phase

**3. Test Subjects**

The component subjected to testing is the search engine from the homepage of www.YYYYY.com, including all filtering logic across appropriate fields.

**4. Pass/Fail Criteria**

Pass criteria:

* Execution of all designed test cases.
* Server response time does not exceed 700 ms.

**5. Entry/Exit Criteria**

Entry criteria:

* The search engine implementation phase is complete.
* A working and configured test environment is available.
* Access to a working and configured virtual machine.

Exit criteria:

* All test cases have been successfully executed.
* The component meets all the established requirements outlined in the attached documentation.

**6. Requirements/Functionalities to be Tested**

All documentation, user stories, scenarios, etc., are attached for reference.

**7. Test Environment**

* Test server: Configured testing server
* Operating system: Windows 10 Home 64-bit
* Browsers included in testing: Firefox, Edge, Opera, Safari

**8. Test Schedule**

Conduct functional tests:

* Verification of functionality based on user stories – 3 hours
* Execution of previously designed test cases – 1 hour
* Verification of backend layer

Conduct performance tests:

* Verification of the average response time
* Determination of the maximum number of requests at which the search engine operates stably

**9. Test Report**

* A list of executed test cases with their statuses
* Performance test measurements
* Other test reports

**10. Tools List**

* JMeter
* TestLink
* Jira
* Browserstack

**11. Incident and Bug Management**

In the testing process, each detected bug must be appropriately reported in the Jira system. This includes:

* Bug priority
* Assigned person (developer)
* Affected component
* Following the organization's workflow, such issues should be fixed by the developer and subjected to retesting.

**12. Roles and Responsibilities**

* Jan Kowalski – Execution of test cases
* Marta Kalinowska – Test case design

**Test Case**

**Test Case: Login Functionality**

**Test Case ID:** TC-001  
**Module:** User Authentication  
**Tested By:**   
**Date:**

**Test Case Description**

Verify that a user can successfully log in with valid credentials.

**Preconditions**

* The application is accessible at [Application URL].
* A test user account exists:
  + Username: testuser@example.com
  + Password: Password123

**Test Steps**

|  |  |  |
| --- | --- | --- |
| Step No. | Action | Expected Result |
| 1 | Open the web application in a browser. | The login page is displayed. |
| 2 | Enter testuser@example.com in the username field. | The entered username is displayed in the username field. |
| 3 | Enter Password123 in the password field. | The entered password is masked (dots or asterisks are displayed instead of text). |
| 4 | Click the "Login" button. | The user is redirected to the dashboard. A welcome message, "Welcome, Test User!", is displayed. |
| 5 | Verify that the browser session contains a valid authentication token. | The token is present in the browser's session storage or cookies. |

**Test Data**

|  |  |  |
| --- | --- | --- |
| Field | Valid Input | Invalid Input |
| Username | testuser@example.com | invaliduser@example.com |
| Password | Password123 | WrongPassword123 |

**Expected Results**

* The user is successfully logged in when using valid credentials.
* An appropriate error message is displayed for invalid login attempts.

**Postconditions**

* The user remains logged in until they manually log out or close the browser.

**Test Status**

* **Pass/Fail:** [To be updated after test execution]
* **Comments:** [Optional comments after test execution]

**Test scenario for automation testing with Selenium**

**Test Scenario: User Login Functionality**

**Scenario ID:** ATS-001  
**Module:** User Authentication  
**Automation Framework:** Selenium WebDriver with [Java]

**Objective**

Automate the login functionality to ensure users can log in with valid credentials and are restricted with invalid credentials.

**Preconditions**

* Selenium WebDriver is set up with the appropriate browser driver.
* Test framework is configured.
* Application URL:
* Test user credentials:
  + Valid:
    - Username: testuser@example.com
    - Password: Password123
  + Invalid:
    - Username: invaliduser@example.com
    - Password: WrongPassword123

**Test Steps for Automation**

|  |  |  |
| --- | --- | --- |
| Step No. | Action | Expected Result |
| 1 | Launch the browser and navigate to the application URL. | The login page is displayed. |
| 2 | Locate the username input field using Selenium (e.g., by ID, Name, XPath). | The username input field is found. |
| 3 | Enter testuser@example.com in the username field. | The entered username is displayed in the field. |
| 4 | Locate the password input field and enter Password123. | The entered password is masked (dots or asterisks). |
| 5 | Locate and click the "Login" button. | The user is redirected to the dashboard, and a welcome message is displayed. |
| 6 | Validate that the page URL matches the expected dashboard URL. | The URL is correct. |
| 7 | Validate that an authentication token is present in the session storage or cookies. | The authentication token is stored. |

**Negative Test Steps for Automation**

|  |  |  |
| --- | --- | --- |
| Step No. | Action | Expected Result |
| 1 | Enter invaliduser@example.com as the username and WrongPassword123 as the password. | An error message, "Invalid credentials. Please try again." is displayed. |
| 2 | Enter blank values for username and password fields. | A validation error, "Username and Password are required," is displayed. |

Automation Code in Java

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class LoginTest {

public static void main(String[] args) {

// Set up WebDriver

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

WebDriver driver = new ChromeDriver();

try {

// Navigate to the application URL

driver.get("http://xxxxxxx-url.com/login");

// Positive Test Case

WebElement usernameField = driver.findElement(By.id("username"));

WebElement passwordField = driver.findElement(By.id("password"));

WebElement loginButton = driver.findElement(By.id("login"));

// Enter valid credentials

usernameField.sendKeys("testuser@example.com");

passwordField.sendKeys("Password123");

loginButton.click();

// Validate login success

String currentUrl = driver.getCurrentUrl();

if (currentUrl.contains("dashboard")) {

System.out.println("Positive test passed.");

} else {

System.out.println("Positive test failed.");

}

// Negative Test Case

driver.get("http:// xxxxxxx -url.com/login");

usernameField = driver.findElement(By.id("username"));

passwordField = driver.findElement(By.id("password"));

loginButton = driver.findElement(By.id("login"));

// Enter invalid credentials

usernameField.sendKeys("invaliduser@example.com");

passwordField.sendKeys("WrongPassword123");

loginButton.click();

// Validate error message

WebElement errorMessage = driver.findElement(By.id("error"));

if (errorMessage.getText().contains("Invalid credentials")) {

System.out.println("Negative test passed.");

} else {

System.out.println("Negative test failed.");

}

} catch (Exception e) {

e.printStackTrace();

} finally {

// Quit the driver

driver.quit();

}

}

}

**Test Summary Report**

**Project Name:** E-Commerce Website  
**Tested By:**  
**Date:**

**1. Test Overview**

The testing phase focused on validating the functionality, usability, and security of the e-commerce website. All test cases were executed in the staging environment to ensure that the application performs correctly before the release to production.

**2. Test Execution Summary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Type | Total Test Cases | Passed | Failed | Blocked | Not Executed |
| Functional Testing | 40 | 36 | 4 | 0 | 0 |
| UI/UX Testing | 10 | 10 | 0 | 0 | 0 |
| Regression Testing | 15 | 13 | 2 | 0 | 0 |
| Security Testing | 5 | 5 | 0 | 0 | 0 |
| Performance Testing | 5 | 4 | 1 | 0 | 0 |

**3. Test Results**

* **Total Test Cases Executed:** 75
* **Total Passed:** 68
* **Total Failed:** 7
* **Total Blocked:** 0
* **Total Not Executed:** 0

**4. Defect Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| Severity | Open Defects | Closed Defects | Total Defects |
| Critical | 1 | 1 | 2 |
| Major | 3 | 4 | 7 |
| Minor | 2 | 6 | 8 |
| Low | 1 | 2 | 3 |

**5. Major Issues Identified**

* **Issue 1:** Dropdown menu fails to display language options on the settings page.
  + **Severity:** Critical
  + **Status:** Open
  + **Notes:** The issue is blocking users from changing their language preference.
* **Issue 2:** Checkout process times out when selecting multiple items.
  + **Severity:** Major
  + **Status:** Closed after fix

**6. Testing Coverage**

* **Functional Testing:** Covered all core features of the website, including user registration, login, product search, cart, and checkout.
* **UI/UX Testing:** Verified the design and responsiveness of key pages on multiple devices.
* **Regression Testing:** Ensured that existing functionality was not broken by recent updates.
* **Security Testing:** Verified the application for common security vulnerabilities (SQL injection, cross-site scripting).
* **Performance Testing:** Tested the application’s load time and response times during peak traffic.

**7. Summary and Recommendations**

* **Overall Status:** The application passed most of the critical functional and security tests.
* **Issues:** Minor issues were identified and are being addressed. However, a **Critical issue** with the dropdown menu needs urgent attention before production deployment.
* **Recommendations:**
  + The critical issue related to the dropdown should be fixed before release.
  + Re-test the checkout process after performance optimizations.
  + The website should be stress-tested under heavy traffic conditions.

**8. Conclusion**

Testing has been mostly successful with a few issues that need resolution. The application is ready for production release once the critical defect is resolved.

**Calendar test with Selenium - code**

public class CalendarTest {

public static void main(String[] args) {

String monthNumber = "6";

String date = "15";

String year = "2027";

String[] expectedList = {monthNumber,date,year};

WebDriver driver = new ChromeDriver();

driver.get("https://yyyyyy.com/");

driver.findElement(By.cssSelector(".react-date-picker\_\_inputGroup")).click();

driver.findElement(By.cssSelector(".react-calendar\_\_navigation\_\_label")).click();

driver.findElement(By.cssSelector(".react-calendar\_\_navigation\_\_label")).click();

driver.findElement(By.xpath("//button[text()='"+year+"']")).click();

driver.findElements(By.cssSelector(".react-calendar\_\_year-view\_\_months\_\_month")).get(Integer.parseInt(monthNumber)-1).click();

driver.findElement(By.xpath("//abbr[text()='"+date+"']")).click();

List<WebElement> actualList = driver.findElements(By.cssSelector(".react-date-picker\_\_inputGroup\_\_input"));

for(int i =0; i<actualList.size();i++)

{

System.out.println(actualList.get(i).getAttribute("value"));

Assert.assertEquals(actualList.get(i).getAttribute("value"), expectedList[i]);

}

driver.close();

}

}