The automated tests were developed using Selenium WebDriver and Java as the primary tools. For the Integrated Development Environment (IDE), Intellij was utilized.

Regarding the testing framework, JUnit in conjunction with Gradle was employed. The intention was to maintain a simplistic structure, with all tests being placed under the "test" folder, and the corresponding objects organized under the "page.objects" folder. The main objective behind using page objects was to segregate the element items from the actual test logic.

Specifically, the test named "LoginParametarized" represents four distinct test cases executed with varying user names. JUnit 5's "ParameterizedTest" library facilitated the execution of these tests with multiple inputs. I preferred this method Instead of using external data from file. For Assertion i used Truth that is a fluent and flexible open-source testing framework designed to make test assertions and failure messages more readable.

Test Cases.

As said before the architecture I followed was to keep all tests below folder test->tests and one class per test case, that way is more organized and clear the structure. Main static configuration values are stored and retrieved from resources-> properties, variables like url, username, password and false password. Under “Base.Test” class BaseTest is handling the properties.

The following scenarios was implemented:

* Authentication/Login
  + Login to Site, happy path scenario that performs login and assert that login was successful.
  + Login Parametrized, using the usernames provided by the site is attempting to login to the site, in every of the 4 scenarios is asserting if login was successful or not.
  + Login Negative, login with wrong password on purpose to verify and assert the error message.
* productsPage
  + Product price, this scenario is retrieving the amount/cost of a product, and then clicks to enter in product details and verifies the amount displayed there.
* Cart
  + Add to cart, this test is retrieving the amount/cost of a product and the name and verifies them after we add it to cart.
  + Remove from cart, in this case we add a product in the cart and then we remove it and verify/assert that there is no other item In the cart
  + Add to Cart and Checkout, on this test I make all the assertions on cart and on checkout and then I calculate the tax amount(8%) and the total amount, make the proper rounding and implement the final assertions on tax amount and total amount before I press the checkout button
* Menu
  + About, this test is opening the menu and selects the about option, there it verifies that cookies popup is displayed and clicks the “accept all cookies” button.
  + Logout, this scenario opens the menu and clicks the logout button, after that it verifies that user is logged out.

In order to create a report with the test results I execute the following command from cmd “gradlew.bat test”. After the execution of the tests, html file is been created with the report. Below a screenshot of a report I generated

A screenshot of a computer

Description automatically generated

To achieve continues integration, so I can execute the tests automatically (for example every night) I used Jenkins and on “**Execute Windows batch command**” I used the command below

“cd\Users\maurogiannopoulos\IdeaProjects\"canvas web lottery" && gradlew.bat test”