Automated test using Selenium Web Driver and C#

1. **Prepare the Environment:**
    - Add the WebDriver executable to your PATH environment variable or place it in the project directory.
    - Ensure all project files, including the AlmaTest.cs class, are accessible.
2. **Install Required Packages in Visual Studio:**
    - Open the project in Visual Studio.
    - Navigate to Tools > NuGet Package Manager > Manage NuGet Packages for Solution.
    - Install the following packages:
        - Selenium.WebDriver
        - Selenium.WebDriver.ChromeDriver
        - Selenium.WebDriver.MSEdgeDriver
        - Selenium.WebDriver.GeckoDriver
        - Xunit
        - Xunit.runner.visualstudio
        - Selenium.Support
3. **Set Environment Variables for Runtime Parameters:**
    - **Variables to configure:**
        - OUS_ID: Observation Unit Set ID to search for (e.g., uid://A002/X639a2a/X2a).
        - BROWSER: Browser to use (Edge, Chrome, or Firefox).
        - PLATFORM: Platform name (e.g., Windows, Linux, macOS).
    - **On Windows:**
        - Open Command Prompt in the project directory and execute:
        - set OUS_ID=uid://A002/X639a2a/X2a
        - set BROWSER=Edge
        - set PLATFORM=Windows
    - **On macOS/Linux:**
        - Add these lines to your terminal:
        - export OUS_ID=uid://A002/X639a2a/X2a
        - export BROWSER=Edge
        - export PLATFORM=MacOS

4. **Run the Test in Visual Studio:**
    - Open Test Explorer in Visual Studio (Test > Test Explorer).
    - Ensure the SearchObservationUnitSetTest is visible in the test list.
    - Select the test and click Run.
    - The browser specified in the BROWSER environment variable will launch, navigate to the Alma Science Archive, and perform the search.
    - If no results are found, the test will fail with the message: *"No observing targets found for the given OUS ID."*
5. **Run the Test from the Console:**
    - Open a terminal in the project directory.
    - Run the test with default settings using:
        - dotnet test
    - Alternatively, set environment variables as described above and then execute:
        - dotnet test
    - If required, ensure the configuration and platform are correctly set by running:
        - dotnet build --configuration Debug --platform "Any CPU"
6. **Test Output:**
    - The test report will be displayed in the console, indicating the test's success or failure.

Case I – using a valid OUS_ID

```
C:\Users\ciungu\source\repos\TestAlma>set OUS_ID=uid://A002/X639a2a/X2a

C:\Users\ciungu\source\repos\TestAlma>dotnet test
Restore complete (0.2s)
  TestAlma succeeded (0.1s) → bin\Debug\TestAlma.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v3.0.1+f8675c32e5 (64-bit .NET Framework 4.8.9282.0)
[xUnit.net 00:00:00.63]   Discovering: TestAlma
[xUnit.net 00:00:00.71]   Discovered:  TestAlma
[xUnit.net 00:00:00.76]   Starting:    TestAlma
Starting Microsoft Edge WebDriver 131.0.2903.48 (eb872b980f9ea5184cec7f71c2e6df8ac30265cc) on port 52307
To submit feedback, report a bug, or suggest new features, please visit https://github.com/MicrosoftEdge/EdgeWebDriver
Only local connections are allowed.
Please see https://aka.ms/WebDriverSecurity for suggestions on keeping Microsoft Edge WebDriver safe.
Microsoft Edge WebDriver was started successfully.
msedgedriver was started successfully on port 52307.
[1737380763.827][WARNING]: This version of Microsoft Edge WebDriver has not been tested with Microsoft Edge version 132.
Test Passed: Found one or multiple observing targets for OUS ID 'uid://A002/X639a2a/X2a'.
[xUnit.net 00:00:13.59]   Finished:    TestAlma
  TestAlma test succeeded (14.7s)

Test summary: total: 1, failed: 0, succeeded: 1, skipped: 0, duration: 14.6s
Build succeeded in 15.2s
```

Case II – using an invalid OUS_ID

```
C:\Users\ciungu\source\repos\TestAlma>set OUS_ID=testAlma

C:\Users\ciungu\source\repos\TestAlma>dotnet test
Restore complete (0.2s)
  TestAlma succeeded (0.1s) → bin\Debug\TestAlma.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v3.0.1+f8675c32e5 (64-bit .NET Framework 4.8.9282.0)
[xUnit.net 00:00:00.58]   Discovering: TestAlma
[xUnit.net 00:00:00.66]   Discovered:  TestAlma
[xUnit.net 00:00:00.71]   Starting:    TestAlma
Starting Microsoft Edge WebDriver 131.0.2903.48 (eb872b980f9ea5184cec7f71c2e6df8ac30265cc) on port 52420
To submit feedback, report a bug, or suggest new features, please visit https://github.com/MicrosoftEdge/EdgeWebDriver
Only local connections are allowed.
Please see https://aka.ms/WebDriverSecurity for suggestions on keeping Microsoft Edge WebDriver safe.
Microsoft Edge WebDriver was started successfully.
msedgedriver was started successfully on port 52420.
[1737380993.398][WARNING]: This version of Microsoft Edge WebDriver has not been tested with Microsoft Edge version 132.
[xUnit.net 00:00:13.22]     TestAlma.AlmaTest.SearchObservationUnitSetTest [FAIL]
[xUnit.net 00:00:13.22]       No observing targets found for the given OUS ID.
[xUnit.net 00:00:13.22]       Stack Trace:
[xUnit.net 00:00:13.22]         AlmaTest.cs(60,0): at TestAlma.AlmaTest.SearchObservationUnitSetTest()
[xUnit.net 00:00:13.25]   Finished:    TestAlma
  TestAlma test failed with 1 error(s) (14.2s)
    C:\Users\ciungu\source\repos\TestAlma\AlmaTest.cs(60): error TESTERROR:
      TestAlma.AlmaTest.SearchObservationUnitSetTest (12s 398ms): Error Message: No observing targets found for the given OUS ID.
      Stack Trace:
         at TestAlma.AlmaTest.SearchObservationUnitSetTest() in C:\Users\ciungu\source\repos\TestAlma\AlmaTest.cs:line 60

Test summary: total: 1, failed: 1, succeeded: 0, skipped: 0, duration: 14.2s
Build failed with 1 error(s) in 14.7s
```

Navigate to code_base_directory/TestResults folder

- Three .trx test report files were added. Those 3 files were generated for the below scenarios
    - Test Passed – an example test using an existing OUS id was executed
    - Test Failed – an id that does not exist was used for one of the search elements
    - Test Failed – an OUS that has no observations was used for the search

7. **The following design patterns and principles are utilized in this test:**
    - **Factory Method Pattern** - The method is responsible for creating and initializing the appropriate WebDriver instance based on the browser type specified in the configuration (Edge, Chrome, or Firefox).
    - **Singleton Deing Pattern** - The WebDriver is instantiated and used in one place to avoid the creation of multiple WebDriver instances during test execution.
    - **Strategy Design Pattern** - The strategy pattern is used where the browser (Edge, Chrome, Firefox) is selected at runtime, allowing the test to be more flexible and support different browsers without modifying the source code.
    - **Wait Pattern** - This pattern helps manage synchronization issues.

8. **Motivating the selected stack (testing framework and language):**
    - **C#** is the latest programming language that we used into the current project for automations.
    - Selenium WebDriver offers excellent support for C#.
    - Extensive NuGet packages (e.g., Selenium WebDriver, browser-specific drivers) simplify dependency management.
    - With .NET Core and .NET 6+, C# code can run on Windows, macOS, and Linux, making it perfect for cross-platform testing scenarios.

    - xUnit is a modern testing framework with minimal overhead.
    - xUnit supports running tests in parallel, reducing overall execution time.
    - xUnit integrates directly with Visual Studio's Test Explorer, enabling easy test discovery, execution, and debugging.
    - xUnit is open source and has a strong community, ensuring regular updates, documentation and tutorials.