Developed by: Feyi Adesanya
Version: 1.0 (2025W)

Tutorial 5: Automated UI testing with Selenium

Welcome to Tutorial 5: Automated UI testing with Selenium! In this tutorial, you will learn about Selenium, an automation testing tool that allows you to control web browsers programmatically. Selenium is used for functional testing, regression testing, and automating web applications.

We will first go through a series of exercises using PHP pages, focusing on handling user interactions, sorting tables, and dynamically loading content. Afterward, we will move on to testing real-world websites using Selenium IDE and recording test.

Time budget for this tutorial: 2 hours

Exercise 0 – Installation

Please refer to the other document provided for this tutorial "Setting Up PHP and Selenium" to install the necessary software

Exercise 1 — Logging into Your Application

New Challenge: Testing `login.php`

Imagine an e-commerce or social media website where users need to log in and be authenticated before accessing the application. This page simulates that functionality, allowing users to log in using a predefined test account (username: `test` | password: `test`). After successful authentication, the user is redirected to the index page.

Goals: You should try and test if logging in works with valid credentials and incorrect login attempts have proper messaging. You should also test if the session is retained and destroyed through logins/logouts.

Exercise 2 — Testing Navigation in Your Application

New Challenge: Testing `index.php`

Think of this as a navigation hub for a web application where users can access different features. The index page contains links to navigate to the main page, hidden page, and an option to log out.

Goals: Try and test that all navigation links work correctly and redirect to the correct pages. You should also ensure logging out takes the user back to the login page, along with verifying broken links and incorrect URLs

Developed by: Feyi Adesanya
Version: 1.0 (2025W)

Exercise 3 — Testing your Application's Data

New Challenge: Testing `main.php`

This page simulates a data management system where users can view, sort, and add entries. It displays a sortable table where users can add new entries.

Goals: Try and verify the sorting functionality and along with adding valid/invalid inputs into the table. You should also verify your data and updates are being saved and persist when refreshing the page.

Exercise 4 — Testing your Applications Delayed/Hidden content

New Challenge: Testing `hidden.php`

This page represents a real-time dashboard. In real life applications elements can load unpredictably due to delays when querying data/computations or issues with servers. The content in this page will take a randomized time to load and will display the data history after a specific number of data points appear.

Goals: Try and create tests that takes into account these delays and wait for indications that content is loaded, and functionality can be testing. Avoid inputting predefined wait times/fixed sleep timers, which can cause unnecessary overhead, and instead create dynamic test (Hint: Selenium's Explicit Waits)

Exercise 5 — Testing the University Website with Selenium IDE

New Challenge: Testing `https://www.mcmaster.ca/`

University websites often contain multiple interactive elements such as login forms, navigation menus, and search functionality. In this exercise, you will use Selenium IDE to record and replay interactions on https://www.mcmaster.ca/

Goals: Try created test recording navigation through key sections such as course registration, student resources, and announcements, verifying all links lead to the expected pages without errors. You may also try entering search queries into the website and verifying the behaviour. Try to find pages with existing issues and run test on the functionality that is passing and failing (Hint: https://gsa.mcmaster.ca/)