**Web Automation and Testing with Selenium**

**Objectives:**

1. Gain proficiency in using Selenium for web automation and testing.
2. Learn to write Selenium test scripts to verify the functionality and reliability of web applications.
3. Understand how to interact with web elements, navigate through web pages, and perform various actions using Selenium.
4. Develop test scenarios for different routes and forms in a web application.
5. Use Selenium tests as a quality assurance tool to identify and prevent issues in web applications.

**Prerequisites:**

1. Basic knowledge of Python programming.
2. Familiarity with web technologies, HTML, and CSS.
3. Understanding of how to install Python libraries using pip.
4. Installation of the Selenium library (pip install selenium).
5. Familiarity with the basics of Flask web applications.

**Introduction:** Testing is an essential aspect of web development. It ensures that web applications work as intended and helps prevent regressions. In this lab, you will explore web automation and testing with Selenium, a popular tool for automating browser interactions. You will work with a Flask web application and create Selenium test scripts to validate its functionality.

**Assignment Setup:**

1. Software Provided: Download and unzip the code base to your respective IDE. It contains routes, forms, and various web elements.
2. Set up a virtual environment (recommended) for your testing environment.
3. Install the required dependencies using requirements.txt or install them manually using pip.

**Instructions:**

**Step 1: Import the Selenium Library**

First, make sure you have Flask and Selenium installed:

bash

pip install Flask

pip install selenium

Then, import the necessary libraries at the beginning of your Python script:

python

from selenium import webdriver

**Step 2: Set Up the Selenium Web Driver**

Choose a web driver compatible with your preferred web browser (e.g., ChromeDriver, Firefox GeckoDriver). Download the appropriate driver in a location that is accessible to your script. Then, create a web driver instance:

python

# Specify the path to your web driver

driver = webdriver.Chrome(executable\_path='/path/to/chromedriver')

Replace webdriver.Chrome with webdriver.Firefox or the appropriate driver class if you're using a different browser.

**Step 3: Navigate to Different Routes**

You can use the get method of the web driver to navigate to different routes in your Flask application:

python

# Navigate to the root URL

driver.get('http://localhost:5000/') # Adjust the URL to match your Flask app's host and port

# Navigate to the search page

driver.get('http://localhost:5000/search')

# Navigate to the form page

driver.get('http://localhost:5000/form')

Make sure to adjust the URLs to match the actual address of your Flask application.

**Step 4: Interact with Web Elements**

If you need to interact with web elements (e.g., filling out a form or clicking a button), you can use Selenium's methods. For example, to fill out a form and submit it:

python

# Find an input element by its name attribute and fill it with data

data\_input = driver.find\_element\_by\_name('data')

data\_input.send\_keys('Selenium Test Data')

# Find a submit button and click it

submit\_button = driver.find\_element\_by\_css\_selector('button[type="submit"]')

submit\_button.click()

In this example, we find an input element by its name attribute and send keys to it. Then, we locate a submit button by its CSS selector and click it.

**Step 5: Perform Assertions and Verifications**

You can use Selenium's capabilities to verify that the correct page is displayed after navigation or that certain elements are present. For example, you can use assertions to check the page title:

python

# Verify that the page title is as expected

assert 'Selenium Tutorial' in driver.title

**Step 6: Close the Web Driver**

After you have completed your interactions and verifications, it's essential to close the web driver to release resources:

python

driver.quit()

Remember to run this script after you've started your Flask application to ensure that the application is accessible. Make sure that your Flask application is running and reachable at the specified URLs before running the Selenium script. Additionally, you may need to adjust your script to match the specific structure and elements of your application.