EE5191 Software Testing & Security Checking

Project 2: Web testing with Selenium

Deadline: Nov. 22, 2023 (23:59)

Requirement:

1. A software under test (SUT) is a public website chosen by a student. No two students may choose the same website. FCFS (first-come-first-select) policy is used in tie-breaking. Registration form is here.
2. Write assertions to evaluate the pass or fail of the test scripts.
   https://www.tutorialspoint.com/what-are-assertions-in-selenium-with-python
3. You should submit the following files 1) Selenium test scripts (.side(if you use IDE) & .py), 2) test report & 3) URL of website (.html). In your test report, you should explain the SUT, your test requirements, your test plan.

Item	Detail	Example
Explanation of the SUT (Website)	Briefly tell me what the website has done.	Website of EE5191: The website offers the syllabus, schedule, teaching materials, including lecture notes and videos, as well as information about projects and midterm presentations.
Test Requirements	Description of what need to be tested in a software system.	Check all the links on the website.
Test Plan	Tell me the details of what you have done and tell me the test input. Related test should be identified in the Test Plan.	Use Selenium WebDriver to find all the links on the website, count the number of <a> tags on the website, and utilize Selenium IDE to test all the links automatically.
Test Result	Tell me the coverage and what bugs were detected with screenshot.	Coverage: Code for calculation of coverage Bug: 2 bugs were found: 1) 7 links have failed. 2) Two images are not displayed correctly.

4. Your score is: (coverage of the test items) x $\log_2$(#html sizes) + 3*(bugs detected)
   coverage of the test items = #checked/(#checked+unchecked)
   Maximum score: 15
   The final score will be arranged if no bugs are found for all of you.
5. Late submission

	Nov. 23	Nov. 24	Nov. 25	Nov. 26	>= Nov. 27
Final Score	80%	60%	40%	20%	0

Before you start, you should know:

1. Selenium is an open-source framework for automating web browsers. It provides a way for software testers and developers to write scripts or programs in various programming languages (such as Java, Python, C#, etc.) to automate interactions with web browsers like Chrome, Firefox, Safari, and Edge. Selenium allows users to simulate user actions like clicking buttons, filling out forms, navigating through web pages, and extracting data from websites programmatically.

   Key components and features of Selenium include:

   1) Selenium WebDriver: WebDriver is one of the core components of Selenium. It provides an API (Application Programming Interface) for interacting with web browsers. Test scripts written using WebDriver can simulate user interactions with a web page.

   2) Selenium IDE: Selenium Integrated Development Environment (IDE) is a browser extension (available for Chrome and Firefox) that provides a record and playback functionality for creating simple test cases without the need for coding. While it's useful for quick test case creation, it's less versatile than WebDriver for complex automation tasks.

2. In Selenium WebDriver,
   There are 3 main function in Selenium WebDriver, which are accessing webpage elements (取得網頁元素), 2) manipulating webpage elements (操作網頁元素) & 3) retrieving the content of webpage elements (取得網頁元素的內容)

   1) Accessing webpage elements (取得網頁元素)

   find_element()

By.ID, id	Find the first matching web element by its id.
By.CLASS_NAME, class	Find the first matching web element by its class.
By.CSS_SELECTOR, css selector	Find the first matching web element using a CSS selector.
By.NAME, name	Find the first matching web element by its name attribute.
By.TAG_NAME, tag	Find the first matching web element by its HTML tag.
By.LINK_TEXT, text	Find the first matching web element by the text of a hyperlink.
By.PARTIAL_LINK_TEXT, text	Find the first matching web element by a portion of the text in a hyperlink.
By.XPATH, xpath	Find the first matching web element using an XPath expression.

   2) Manipulating webpage elements (操作網頁元素)

click()	element	Click the left mouse button.
click_and_hold()	element	Click and hold the left mouse button.
double_click()	element	Double-click the left mouse button.
context_click()	element	Right-click (requires specifying element location).
drag_and_drop()	source, target	Click the source element, move to the target element, and release.
drag_and_drop_by_offset()	source, x, y	Click the source element, move to the specified

		coordinates, and release.
move_by_offset()	x, y	Move the mouse cursor to a specified position.
move_to_element()	element	Move the mouse cursor to a specific element.
move_to_element_with_offset()	element, x, y	Move the mouse cursor to a relative position within a specific element.
release()	element	Release the mouse button.
send_keys()	values	Send specific keyboard key values.
send_keys_to_element()	element, values	Send keyboard key values to a specific element.
key_down()	value	Press a keyboard key.
key_up()	value	Release a keyboard key.
pause()	seconds	Pause for a specified number of seconds.
perform()		Execute the stored actions.
back()		Navigate backward in the browser's history
refresh()		Refresh the current webpage

3) Retrieving the content of webpage elements (取得網頁元素的內容)

text	The text content of the element.
get_attribute	The value of a specific HTML attribute of the element.
id	The id attribute of the element.
tag_name	The tag name of the element.
size	The dimensions (width and height) of the element.
location	The locations (x-axis and y-axis) of the element.
screenshot	Capture a screenshot of the element and save it as a PNG image.
is_displayed()	Check if the element is visible on the webpage.
is_enabled()	Check if the element is enabled (i.e., interactable)
is_selected()	Check if the element is selected (e.g., in the case of checkboxes or radio buttons).
parent	The parent element of the element.

Example is given by https://steam.oxxostudio.tw/category/python/spider/selenium.html & https://selenium-python.readthedocs.io/locating-elements.html

6. You can use Selenium WebDriver or IDE. Try your best to test all the links or buttons on the website.
7. In IDE (chrome extension), you must write assertion / verification to check pass / fail in the test scrpit. Example is given in zip file (hkgov.side & test_hkgov.py). After recording, .side and .py can be generated.
8. In WebDriver, you also need to write assertion / verification to check pass or fail. Example is given in zip file (project2_1.py & project2_2.py)
9. **You should provide the code to prove the coverage.** You can refer project2_2.py.

```
The number of <a> : 82
The number of <a> (checked) 75
The number of error in <a> : 7
The number of <a> (unchecked) : 7
```

OR project1_1.py to check no. of tags first, then calculate no. of assert / .click() in .side or .py