Lecture 33
Selenium - Part II

ECE 422: Reliable and Secure Systems Design

Instructor: An Ran Chen
Term: 2024 Winter

Schedule for today

- Demo: Cookie Clicker
  - Step 4: Assertion
- What more can we do with Selenium?
- Surveys, feedbacks and ideas
  - SPOT Survey
  - ECE 422 course website
- Final exam
  - Format and type of questions
  - Review sessions

Demo: Cookie Clicker

Cookie Clicker is a 2013 web game designed to have users click on a big cookie on the screen.

- Steam version released in 2021, reaching top 15 of Steam games at the time

Build an automation script for clicking the cookie 100 times

- Input: Cookie Clicker URL
- Learning Objectives: Click + Wait + Scrap + Assert
- Output: Assert that the cookie has been clicked 100 times
- Source code available on GitHub: automation script

Demo guide (Python)

Step 1: Installation

- Python language bindings for selenium package

Step 2: Python script setup

- Getting started with WebDriver

Step 3: Navigate and click

- Interacting with the page
- Explicit Waits

Step 4: Assertion

- Test case with assertions

Step 1: Installation

Step 1: To run Selenium, we first need to install a WebDriver.

- WebDriver is responsible for managing the content interactions, without requiring browser-specific code

Every browser provides WebDriver supports:

- Chrome: ChromeDriver
- Firefox: GeckoDriver
- Edge: Microsoft Edge WebDriver
- Safari: SafariDriver

If you are using Chrome version 115 or newer, check the Chrome for Testing availability dashboard

Note that this demo uses: chromedriver for mac-x64 (click here to download)

Step 2: Python script setup

main.py available

Step 2: create a Python script (e.g., main.py) in the same folder as the WebDriver

- main.py is the automation script we use for clicking the cookie

Set up the script by importing the WebDriver, then try to launch the browser:

- Import the WebDriver and By class (for locating elements)

```
from selenium import webdriver
from selenium.webdriver.common.by import By
```

- Create an instance of Chrome WebDriver

```
driver = webdriver.Chrome()
```

- Use driver.get method to navigate to a given URL

```
driver.get("https://orteil.dashnet.org/cookieclicker/")
```

Step 3: Navigate and click

Step 3: Select a language and click on the cookie

- Step 3.1: Select the English element from language popup
  - Appear on first visit (incognito for test case rerun)
  - Take time to load (wait for the element to appear)
- Step 3.2: Click on the cookie
  - Take time to load (wait for the element to appear)
  - Click 100 times (add loop)

Step 4: Assertion

main.py available

Step 4: Assert that the cookie is less than 100, after an upgrade

Step 4: Assertion

main.py available

Step 4: Assert that the cookie is less than 100, after an upgrade

- Upgrade for cookie auto-generation

```
grandma_xpath = "//*[@id='product1']"
grandma_element = driver.find_element(By.XPATH, grandma_xpath)
grandma_element.click()
```

- Capture count element again and assert that it is less than 100
  - To avoid elements become stale, it is essential to refresh the reference
  - If the assertion fails, then the script throws an AssertionError

```
count = driver.find_element(By.XPATH, count_xpath).text.split(' ')[0]
assert int(count) < 100
```

What more can we do with Selenium?

- Scrape data from websites
  - Python, Selenium WebDriver + pandas
- Automated testing
  - Java, Selenium WebDriver + TestNG
  - Three-steps process:
    - Set up the test data @BeforeTest
    - Perform a set of actions @Test
    - Evaluate the results @AfterTest
  - Tests are often about the "happy path"
    - General works (no corner cases)
    - Encouraged behaviors for Selenium tests

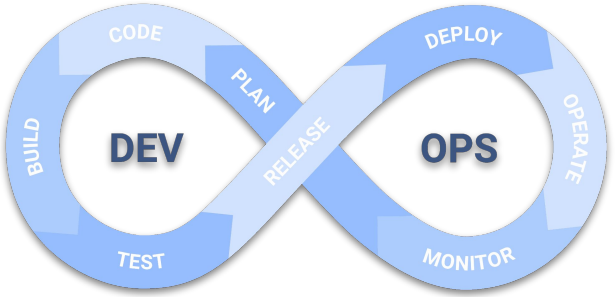
```
public class LoginTest {
    @BeforeTest
    public void setUp(){ … }

    @Test
    public void login(){ … }

    @AfterTest
    public void tearDown(){ … }
}
```

Sample automated test: Login

```
@Test
public void login(){
    driver.get("https://example.com/login");
    WebElement username = driver.findElement(By.id("user_email"));
    WebElement password = driver.findElement(By.id("user_pass"));
    WebElement login = driver.findElement(By.name("submit"));
    username.sendKeys("bob@example.com");
    password.sendKeys("bob_password");
    login.click();
    WebElement welcome = driver.findElement(By.xpath("//span[@name=\"welcome_text\"]"));
    Assert.assertEquals(welcome.getText(), "Welcome Bob")
}
```

Selenium in DevOps

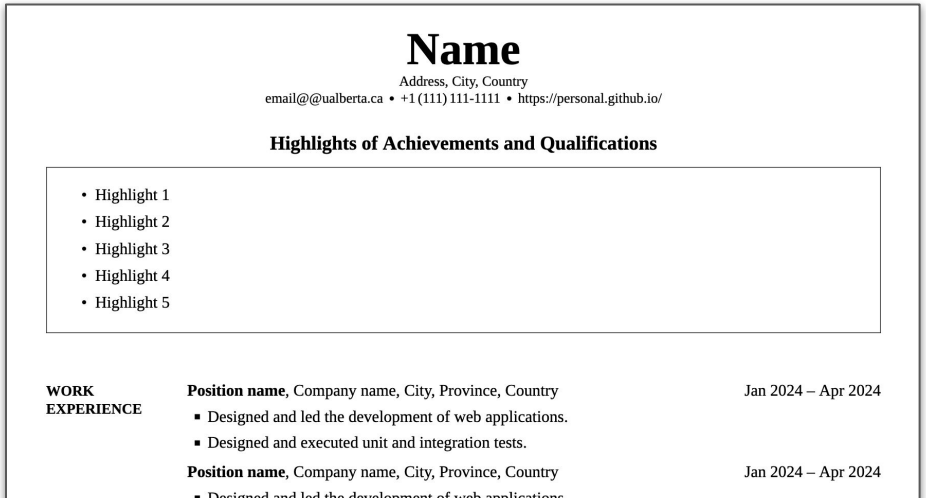
Selenium can be integrated into DevOps workflow to encourage collaboration:



- For Dev (developers)
  - Continuous Integration (CI)
    - Automatically run tests when code changes occur
  - Continuous Deployment (CD)
    - Validate functionality before pushing changes to production
- For Ops (operators)
  - Parallel Test Execution
    - Speed up the test execution time with cross-browser and cross-platform testing
  - Monitoring and Alerting
    - Continuous testing in production and staging environment to monitor the application

CV Template

- CV template available on Overleaf (written in LaTeX)
- Highlights of Achievements and Qualifications
  - Be specific and concise in your experiences, achievements and qualifications
  - E.g., Bachelor degree in Software Engineering with 5 years of Object-Oriented Programming experience
- Interview tips (Undergrad.)
  - Competence based on experience
  - Character as a coworker
  - Practice speaking points (bullet points)
  - Discuss career direction
    - Hint on no relocation
    - Hint on dream job



Name

Address, City, Country
email@@ualberta.ca • +1 (111) 111-1111 • https://personal.github.io/

Highlights of Achievements and Qualifications

- Highlight 1
- Highlight 2
- Highlight 3
- Highlight 4
- Highlight 5

WORK EXPERIENCE

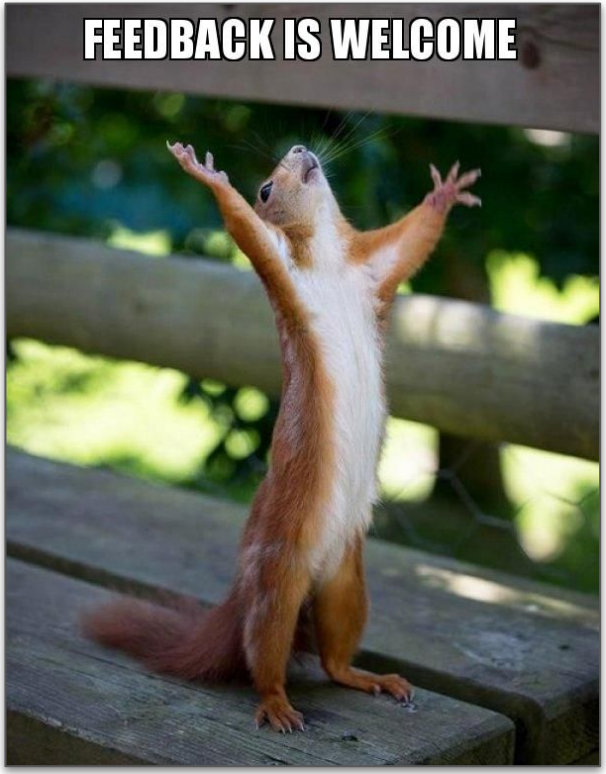
Position name, Company name, City, Province, Country — Jan 2024 – Apr 2024
- Designed and led the development of web applications.
- Designed and executed unit and integration tests.

Position name, Company name, City, Province, Country — Jan 2024 – Apr 2024

Schedule for today

- Demo: Cookie Clicker
  - Step 4: Assertion
- What more can we do with Selenium?
- Surveys, feedbacks and ideas
  - SPOT Survey
  - ECE 422 course website
- Final exam
  - Format and type of questions
  - Review sessions

SPOT Survey



SPOT Survey: Click here

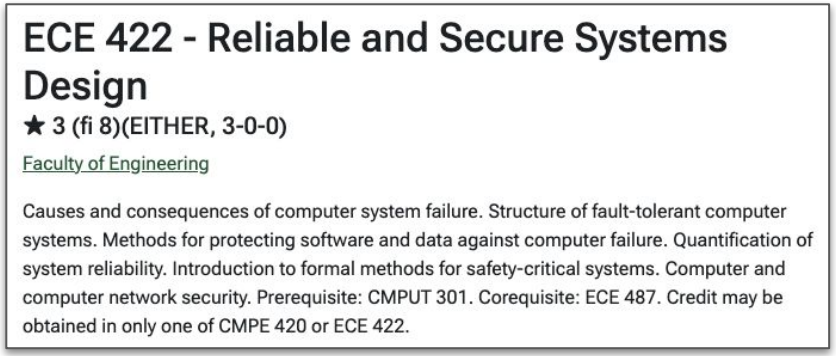
- 15 minutes
- Course number: 15754
- Survey close: Apr 14, 11:59 PM

“Beginning with winter 2024 courses, instructors are now required to provide 15 minutes of class time to complete the survey. Please note that instructors will not be present during this class time.”

ECE 422 course website

Why? Course archive for past students (you) + Preview for future students:

- Syllabus + Lecture slides
  - Module 1: DevOps + SREs
  - Module 2: Reliable and Fault-Tolerant Design
  - Module 3: Security Principles
  - Module 4: Emerging topics - Bitcoin and Selenium
- Demos + Course projects
  - Web Security: XSS, CSP, SOP
  - Selenium: Cookie Clicker
- Midterm and final exam



ECE 422 - Reliable and Secure Systems Design

★ 3 (fi 8)(EITHER, 3-0-0)

Faculty of Engineering

Causes and consequences of computer system failure. Structure of fault-tolerant computer systems. Methods for protecting software and data against computer failure. Quantification of system reliability. Introduction to formal methods for safety-critical systems. Computer and computer network security. Prerequisite: CMPUT 301. Corequisite: ECE 487. Credit may be obtained in only one of CMPE 420 or ECE 422.

ECE 422 course website

How can you help? Any feedbacks are welcome!

- Email me (anran6@ualberta.ca)

Example of feedbacks and ideas:

- Available content for both past or future students
  - E.g., Logs/blogs for new content updates
- Website designs
  - E.g., Hidden content when you know how to reverse-engineering a hash function
- Interactive demos
  - E.g., Hosting a server to have students try XSS

Schedule for today

- Demo: Cookie Clicker
  - Step 4: Assertion
- What more can we do with Selenium?
- Surveys, feedbacks and ideas
  - SPOT Survey
  - ECE 422 course website
- Final exam
  - Format and type of questions
  - Review sessions

Format of exam

- Classroom: The Centennial Centre for Interdisciplinary Science (CCIS) 1-160
- Date: Wednesday, Apr 24th, 2024
- Duration: 2 hours
- Closed book
- Materials: all materials posted in the lecture slides
    - => 60% on new materials, < 40% on old materials
- The final counts for 30% of the overall course grade

Email me if you need to know your grade early for work permit or graduation (estimated early grade release date: ~ May 1).

Type of question

- True/False
- Multiple choice
- Short answer
- Essay questions
  - Explain and give examples
- Computational questions
  - Bring your own calculator

Course registration system

User story: As a security admin, I want to ensure the integrity of the data

- Lecture 13: The CIA Triad
- Lecture 14: Hash function and Digital Signature
    - Hash collision on integrity checks

Key concepts:

- Digital Signature: Authenticity + Integrity + Non-repudiation
- Difference between hashing and encryption: irreversibility

Course registration system

User story: As a security admin, I want to ensure the confidentiality of the system.

- Lecture 15: Authentication
  - Multi-factor authentication
- Lecture 16: Access Control
  - Models of access control
- Lecture 17: Encryption
  - Symmetric and asymmetric encryption

Key concepts:

- MFA: knowledge, possession, biologic, location and time
- DAC, RBAC, MAC, ABAC
- Asymmetric: encryption = sender’s private key; decryption = public key

Course registration system

User story: As a web admin, I want to prevent race condition for course registration.

- Lecture 20: The Dining Philosophers Problem
  - Race condition
  - Atomic locking

Key concepts:

- Locks to prevent multiple users from registering the courses at the same time
- Atomic locks for course reservation