Assignment 4

System Testing

Introduction

In this assignment, you will create acceptance, end-to-end tests for a Web-based open-source application.

The assignment has the following steps:

1. Finding partners
2. Use the template repository
3. Choose software and some user stories to test
4. Submit your choices
5. Test the software using behavior-driven testing (Cucumber)
6. Test the software using model-based testing (Provengo)
7. Generate test suites for your model (Provengo)
8. Write a report
9. Submit your assignment 🙂

Make sure you **read the entire assignment** before starting. Pay attention to the **General Implementation Tips** section **BEFORE** you begin Step 5; it will help you.

Goals

The goals of this assignment are:

- Experience the use of behavior-driven testing (i.e., Cucumber)
- Experience the use of model-based testing (i.e., Provengo)
- Understand the difference between the two types and the advantages of each one.
- Practice with modeling system behavior.
- Practice CTD coverage techniques.

1. Finding partners

The assignment is submitted in pairs of two groups. Find another group that you want to work with as soon as possible.

2. Use the template project

Join our GitHub Classroom zone by following these steps:

1. Enter the classroom link
2. Choose your name from the list.
3. You will now need to choose/create a team. Your team name is <group 1 number>-<group 2 number>. For example, if the two groups are 01 and 02, your team's name will be 01-02.
   Search for your team name. If you've found it - join it. Otherwise, create it.
4. Press accept, and you are done. Now, you will have a new repository with access rights to your team.

The repository has three directories: Selenium, Cucumber, and Provengo. There are four README.md files:

a. README.md: A general description of the repository

b. Cucumber/README.md: A description of your model-based tests

c. Provengo/README.md: A description of your model-based tests

d. Provengo/helloprovengo/README.md: A short user manual for the Provengo tool

e. Selenium/README.md: A general description for running the Selenium server

Updating these files (except for **d** and **e**) is part of the assignment (detailed below). Take a look at them before Step 5.

3. Choose software user stories to test

You will test one of the following web-based applications: Moodle, OpenCart, or PrestaShop.
For each of these applications, we prepared a list of possible projects, each has two user stories.
You should choose a project and write the group numbers in the relevant lines.

4. Submit your choices

Each group should submit a ".txt" file to the assignment page in Moodle. The text file should have the following structure:

```
Group 1 number: <group number>
Group 2 number: <group number>
Group 1 IDs: <a comma-separated list of ID numbers>
Group 2 IDs: <a comma-separated list of ID numbers>
Repository URL: <the URL of your repository>
Selected use-case number: <the line number is the spreadsheet of your use case>
```

IMPORTANT: Don't wait for this step till the deadline. You should do it once you select a use case and create the repository.

5. Test the software using behavior-driven testing (Cucumber)

In this part, you will test your module using Cucumber.

Prerequisites:

a. You will need to make sure that the Selenium Server is working. See Selenium/README.md for more details.

See "Cucumber/README.md" for a detailed description of this part.

6. Test the software using model-based testing (Provengo)

In this part, you will create a test model for the two use cases and connect them to the system.

In addition, you will define two coverage criteria for creating a test suite:

- *Domain-specific criterion:* A criterion that is relevant to your specific user stories. See the Ranking and Prioritization of Test Cases lesson for more information.
- Two-way/pairwise criterion: A criterion that tries to cover all possible orders of any two events. See the Creating and Running Test Suites lesson for more information.

Finally, you will use these criteria to create two test suites and thoroughly test the system (and the model).

Note, if some of the tests fail - you need to check whether it is a problem with your model (and therefore, you need to fix it and start the process again) or if it is a bug (and then you should add it to your report).

Prerequisites:

a. You will need to make sure that the Selenium Server is working. See Selenium/README.md for more details.

b. You will need to install Graphviz to generate the graphs' PDF.

c. Make sure every js file inside the spec/js directory has the following first line: /* @provengo summon selenium */

d. The following links will help you with this part:

   i. CLI documentation

   ii. Provengo course

See "Provengo/README.md" for a detailed description of this part.

7. Write a report

Update all README.md files (except for **d** and **e**, see Section 1). Specifically, replace all *$$*TODO*…$$* according to the instructions inside the $$.

General Implementation Tips

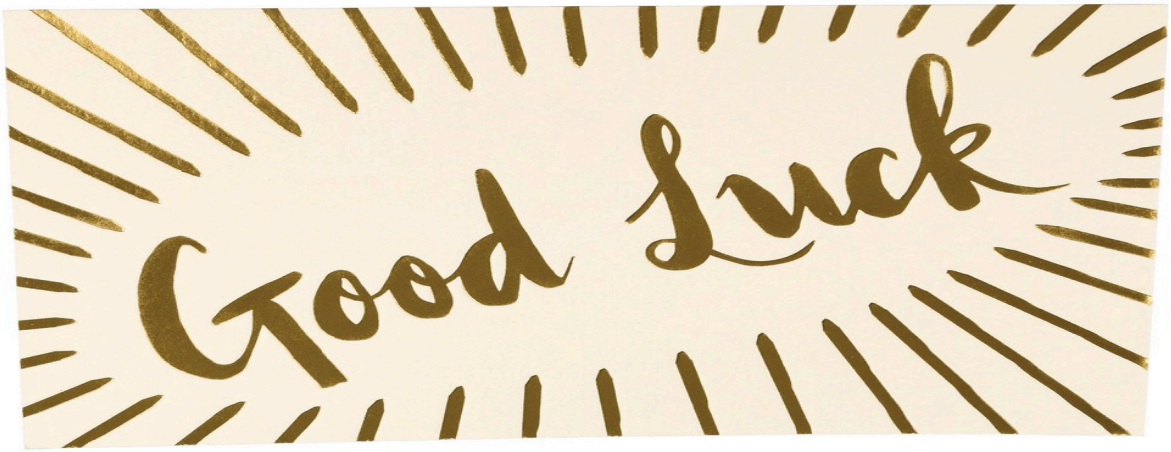
a. In acceptance testing, resetting the data every time you run the tests may be problematic. In Moodle, for example, to automatically delete all courses/quizzes before each test, you would need to use several selenium commands that may also fail (and it will take you some time to develop this functionality. There are several ways to overcome this problem:
   a. Manually delete the course. While it is generally unacceptable, it is okay in this assignment.

   b. Create a "Before" function/script that deletes the course using selenium. See Provengo's documentation to see how to call this function/script.

b. Step 5 and Step 6 test the same user stories and screens. Therefore, you will use the same xpaths for both steps.

8. Submit your assignment

There is no need to submit the assignment. We have access to your repository, and once you submit your choices, we know how to evaluate your work.


Good Luck