Testing Learning

Workgroup number: E8.01

Repository: https://github.com/javiervaz01/Acme-Toolkits

Date: 2022/05/26

Name	Corporate e-mail
Carrasco Núñez, Alejandro	alecarnun@alum.us.es
Durán Terrero, Andrés	anddurter@alum.us.es
López Benítez, Pablo Delfín	pablopben@alum.us.es
Núñez Moreno, Pablo	pabnunmor@alum.us.es
Robledo Campa, Pablo José	pabrobcam@alum.us.es
Vázquez Monge, Francisco Javier	fravazmon@alum.us.es

Table of Contents

Executive summary

The following document consists of an explanation of what we know about testing a Web Information System after taking the subject of Design and Testing II, from this degree of Software Engineering.

Revision table

Revision number	Date	Description
v1	2022/05/26	Initial version
v2	2022/05/27	Second version

Introduction

It has already been taught to us in this course the importance of testing a Web Information System, not only because of making sure that our code is right, but for keeping it simple, eliminating defects or guaranteeing quality.
We have learned that we can envision testing from 2 approaches: a functional and performance way.
The **functional tests** detect failures by comparing the actual results of a request to its expected results.
On the other hand, with the **performance testing**, we can check if performance data deviates from the expectations. This approach requires performing statistical analysis at a given confidence level.

Functional Testing

When testing a Web Information System we had to face different concepts:
- **SUT** (System under test). In the general theory of testing, the SUT ranges from a method in a class to a full system. Our application, **Acme Toolkits** is the SUT, and we test it feature by feature.
- **Fixture**. It consists of a web of artefacts that will be needed to set up for the SUT to be tested. The fixture ranges from an object to a full system. Our **database** is the fixture.

When implementing the tests we had to make sure that we had fulfilled the 3 cases that we were taught in classes:

1) **Positive test cases**. It is a script that checks that the SUT works well in a context in which it is expected to work well. Working well means that the SUT does not issue any errors or exceptions and produces the expected results.
For instance, we implemented a positive test case from the feature any.chirp, in which any kind of user (authenticated and not authenticated), could create a chirp successfully.
This kind of test has been the most habitual for us to implement in previous subjects.

2) **Negative test cases**. It is a script that checks that the SUT works well in a context in which it is expected to report an error.
Furthermore, in order to implement a complete negative test, we have learned how to use invalid data (trying to submit it) and we have taken into account some pieces of advice that we will use in the future, such as, empty forms (attribute containing no character) or binding errors (when the user enters datum that cannot be parsed). However, the most interesting ones for us were errors that involved patterns, ranges of dates and complex classes, given that we had never implemented a negative test that was similar to reality.
For instance, we implemented a negative test case from the feature any.chirp, in which any user (authenticated or not authenticated), tried to create a chirp with wrong input values. Therefore, the chirp could not be created and the corresponding errors were displayed in the user interface.

3) **Hacking test cases**. It can also be considered as a negative test case. In this context, it is a script that checks that the SUT works well in a context in which it is expected to panic. For example, the context where a user requests a feature with the wrong user role.
Regarding these kinds of tests, for most of us, we had never paid much attention to them in previous subjects, so they were really interesting for us here.
For instance, we implemented a hacking test from the feature administrator.systemconfiguration, in which an anonymous user, an inventor and a patron tried to show the system configuration, which can only be accessible by the administrator. The application did not allow these features to be shown for the forbidden roles.

Regarding the organisation of the tests, we applied the techniques used on the Acme Jobs project.
We created **Test classes**, which are collections of related test cases that are encapsulated in a single class. Our test classes encapsulate positive, negative and hacking test cases (if any) regarding each particular feature.

Our approach to functional testing is called **E2E** (end-to-end). It attempts to test software "naturally" since the goal is to simulate a user interacting with the web interface, while checking that the system performs as expected.
The test cases consist of scripts that use the Gecko driver to simulate a user interacting with Firefox (clicking on options, buttons, and submits) and reading the results returned by your application to check that they are the expected ones.

Performance Testing

We believe that the performance testing has been one of the highlights of the testing part, given that this was the first document of that kind that we made.
Firstly we ran our test suites and collected performance data, thanks to the excel provided by the generated logs. Then, we compute the **confidence interval** for the mean of the wall time (we used the standard of 95%).
However, the goal was to prove that the performance could be improved, by means of a reforization of code (using the computer of a different teammate). Then, we repeated the collection of performance data with the new information.
Finally, we launched the **Z test** to analyse both examples (before and after refactoring code).
Thanks to the p-value, resulting from this analysis, we could conclude if the performance has been improved.

Conclusions

After having taken this subject we believe that we have acquired knowledge about testing a Web Information System, that we did not have before. Most of us had never implemented a negative or a hacking test case, and now we consider them a very useful approach.

We hope to have achieved the expected goals regarding testing, however we seek learning more. Therefore, we hope to become real experts in the field of testing and learn everything about it, not only the basics, in our future.

Bibliography

Intentionally blank.