**Software Quality Assurance (SQA)**

**PREPARED FOR**

Frank Coenen

**PREPARED BY**

Maarten Blömer

Nick Krijgsman

Faruk Aydin

Jursley Gonzalez

Vincent Andersen

# **Table of Contents**

[**Table of Contents**](#_pykfpgqrlej) **2**

[**Case Study**](#_3s39dys3cx66) **3**

[Case goal:](#_qwqn8mh5s18) 3

[Case questions:](#_4frtw1206ej) 3

[Questions:](#_rhg0mb87rhxw) 3

[Automated Testing](#_j9mmpthv5j57) 3

[**Brainstorm**](#_whir32t8ue2v) **4**

[**Individual Research**](#_wisaxg2eeii0) **5**

[Long List](#_bnr4vo3zzw77) 5

[Short List](#_pcvo6hkru4c8) 7

[**Plan**](#_j4m0m4i7c598) **8**

[**Recommendation**](#_ortrbpg268ws) **9**

[Postman](#_3szxip10x72h) 9

[Postman IDE](#_fpzyyaqgwg2a) 9

[Newman](#_i6lt28bzwc69) 9

[Selenium Framework](#_gz49jesp4trp) 9

[Selenium IDE](#_mjrrjhdr32vk) 9

[Selenium Webdriver](#_g6e0vadqnqz) 9

[**Prototypes**](#_vrxztax8jlwa) **11**

# Case Study

## **Case goal:**

The goal of this case study is to get more familiar with Software Quality Assurance tools and frameworks. By the end of this case we have described two different tools and listed their pro’s and con’s. This will result in a recommendation on which tool to use on our platform.

## Case questions**:**

### Questions**:**

* What is automated acceptance testing and how can it be used to reduce the amount of discussion with the PO about the acceptance of user stories?
* What tools/frameworks are available for our development platform?
* Which tools are available for testing code coverage and static code analysis?

## Automated Testing

An automated acceptance test is an essential component of a continuous delivery style testing strategy.

An acceptance test is an executable specification of the behaviour of the system. These tests give important and different insight into the behaviour of a system. These tests evaluate the system from the perspective of an external user, they also provide us with an automated definition of done.

By the use of different tools like Selenium, Katalon Studio and JaCoCo we can automate these tests, instead of manually for example having to go through the different functionalities web application.

The amount of discussion with the PO gets reduced by:

* Decreasing the chance and severity of new issues and regressions.
* Providing a clear definition of done for the PO, developer and customer/user on how the system should behave.
* Issues that could be encountered can be encapsulated and tested during every release cycle.
* Automation tools generate reports that can be shared with the PO.

# **Brainstorm**

# **Individual Research**

## Long List

|  |  |
| --- | --- |
| **Tool/Framework** | **Tool Type** |
| [Katalon Studio](https://www.katalon.com/) | All-in-one test automation solution for Web, API, mobile and desktop |
| [Selenium Framework](https://www.selenium.dev/) | All in one test automation solution |
| [JUnit](https://junit.org/junit5/) | Unit testing framework for Java |
| [JaCoCo](https://www.eclemma.org/jacoco/) | Code coverage library for Java |
| [SoapUI](https://www.soapui.org/) | API Testing tool |
| [LGTM](https://lgtm.com/) | Code analysis platform |
| [TestNG](https://testng.org/doc/) | TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc... |
| [SonarQube](https://www.sonarqube.org/) | Static code analysis |
| [CheckStyle](https://checkstyle.sourceforge.io/) | Static Code Analysis Tool for checking if Java source code is compliant with specified coding rules. |
| [SpotBugs](https://spotbugs.github.io/) | Static Analysis Tool to look for bugs in Java code. |
| [REST Assured](https://rest-assured.io/) | REST Assured is a Java DSL for simplifying testing of REST based services built on top of HTTP Builder |

# 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pugh Matrix** | **Weight** | **Katalon studio** | **JaCoCo** | **Selenium Framework** | **TestNG** | **JUnit** |
| **Ease of developing and maintaining the scripts** | 5 | + | + | + | + | + |
| **Ease of test execution for technical user** | 5 | + | + | + | + | + |
| **Support to web, desktop and mobile application** | 5 | + | + | + | + | + |
| **Intuitive test report generation** | 4 | + | + | + | + | + |
| **Cross browser testing** | 4 | + | - | + | - | - |
| **Support to keyword and data driven testing** | 4 | + | - | + | + | + |
| **Technical support and assistance** | 3 | + | + | + | + | + |
| **Java Language support** | 3 | + | + | + | + | + |
| **Pricing** | 3 | - | + | + | + | + |
| **Devops integration with build** | 3 | + | -/+ | + | + | + |
|  | total + | 36 | 26 | 39 | 35 | 35 |
|  | total - | 3 | 9 | 0 | 4 | 4 |
|  | total | 33 | 17 | 39 | 31 | 31 |

## Short List

We have chosen 2 tools based on the the following criteria:

* Java language support
* Ease of developing and maintaining the scripts
* Cross Browser Testing
* Pricing
* Intuitive test report
* Integration with CI/CD tools

Our development team has had a brief experience with testing tools before, some of which are present in the list. In order to determine which tool or framework suits our needs we will be comparing them to our discussed previous experiences with some of them and compare and contrast them. Based on that we will be choosing the following 2 tools and build prototypes for them.

|  |  |
| --- | --- |
| **Tool/Framework** | **Tool Type** |
| Selenium | UI |
| Postman | Backend API Tester |

## 

# **Plan**

Most CI/CD tools have an email function integrated. So when the system is tested a report is generated and sent to the people listening to that branch. To prevent the product owner from being spammed with emails about every push and every test, we suggest to use a staging branch. This staging branch is the place where everything is tested one last time before it goes to production. This is the place you want to involve your product owner, as he is the one to decide if the system is inline with his demands. By sending an email with the test results, static code analysis and code coverage report to the production owner at this point, we involve him and he is up to date about the functionality of the system.

The User Acceptance Tests (UAT) can be partially done by the automated tests in the pipeline, and thus a report can be sent to the product owner. A tool that can be useful to use is Selenium. With this tool, or the same kind of tools, you can test the UI’s functionality. However some tests can not be done automatically. As most software solutions have to be easily learnable, efficient to use, and aesthetically pleasing for it to deliver its desired benefit. These are subjects that are (near) impossible to automate, due to being subjective matters. To keep the product owner involved in these kinds of user stories it is a good practise to do UX/UI research, discuss designs with end users and, if necessary, involve the product owner in certain UI decisions.

All and all if the testing is set up correctly and certain terms are set with the product owner, these tests can reduce the discussion time on whenever a user story is accepted or not, as the product owner should have the information at hand.

# **Recommendation**

After prototyping with the chosen frameworks we have come to the following conclusion regarding the recommendation of both of them:

|  |  |  |
| --- | --- | --- |
| **Tool** | Postman | Selenium Framework |
| **Recommendation** | **+** | **=** |

## Postman

### Postman IDE

In the Postman IDE all API related tests can be written and executed. These tests can be exported into a json file and added into your project folder to automate the tests in the CI/CD pipeline.

### Newman

Using the generated json files from Postman IDE, the tests will automatically run after each commit to your assigned branch in your CI manager.  
Our recommendation is to set this up with TravisCI, it makes it easy to read the result logs from the executed tests. Travis will also automatically send an email confirming the result of the tests. This is an easy way to communicate to the PO about the status of the project. Automatic forwarding can be turned on for when tests run successfully or fail.

## Selenium Framework

### Selenium IDE

The Selenium IDE has a quick and easy way to make tests without knowing code. The UI is easy to follow and anyone without testing experience will be able to make a plethora of tests that can all be run with a single click. The addition of it being real time has an addition of being more convincing to the PO as that person will be able to see the tests being run.

### 

### Selenium Webdriver

The Selenium Webdriver is a lot more complicated than the Selenium IDE as it requires some unorthodox setup before you can make tests. The idea itself is more or less the same as Selenium IDE, except that you will have to write the steps in code, which requires more time as you will have to find out the website code structure.

In terms of information and integration, it was surprisingly hard to find good examples that would use TravisCI. Although other CI/CD examples were found, because of the setup and nature of CI/CD platforms, it will always be a bit different and we couldn’t figure out the correct setup needed to run the pipeline correctly.

Ultimately we can recommend Selenium IDE wholeheartedly due to its ease of use and low entry barrier. The same cannot quite be said about Selenium Webdriver as finding the correct setup within our timespan was quite tedious. Locally however, the test was set up and ran quite well as expected along with documentation that is simply okay.

While we do think it may be possible, we will be resuming study in our research document as we would like to use Selenium.

# 

# Prototypes

<https://github.com/MaartenBl/S63-1Hackaton/tree/selenium-prototype>

<https://github.com/MaartenBl/S63-1Hackaton/tree/postmantest>