January 11, 2023

QA Testing Engineer Profile Test (Behave, Selenium, SQL, Python, testing knowledge)

By Cristina Henao Osorno

This test was built in 3 parts, a functional one with requirement analysis, automation, SQL Databases basics and some questions about software testing.

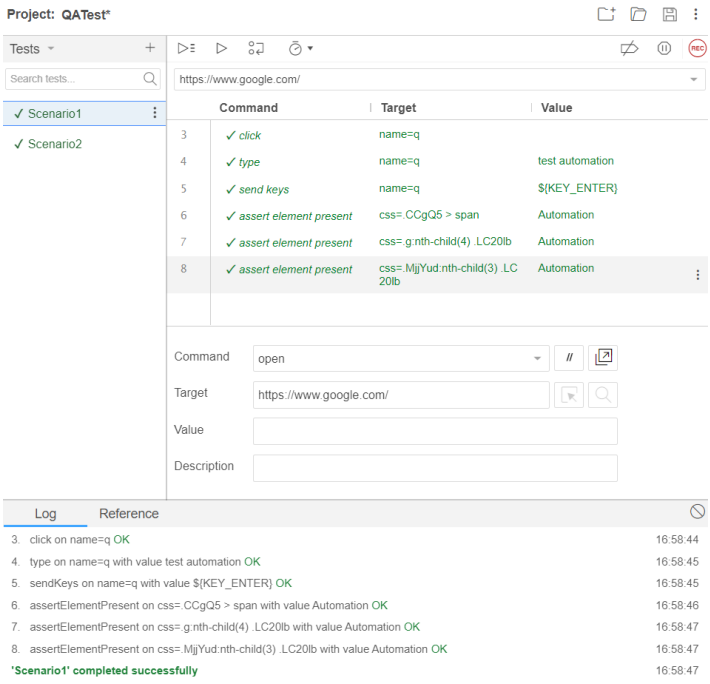
Part 1:

With the following scenarios, automate the interaction. For this you should use whether version of the listed frameworks you like: Python and selenium.

a) *Scenario 1: User can search with "Google Search"*
   - Given I'm on the homepage
   - When I type "test automation" into the search field And I click the Google Search button
   - Then I go to the search results page, and the first 3 results contain the word "automation"

Figure 1

Automation on Selenium

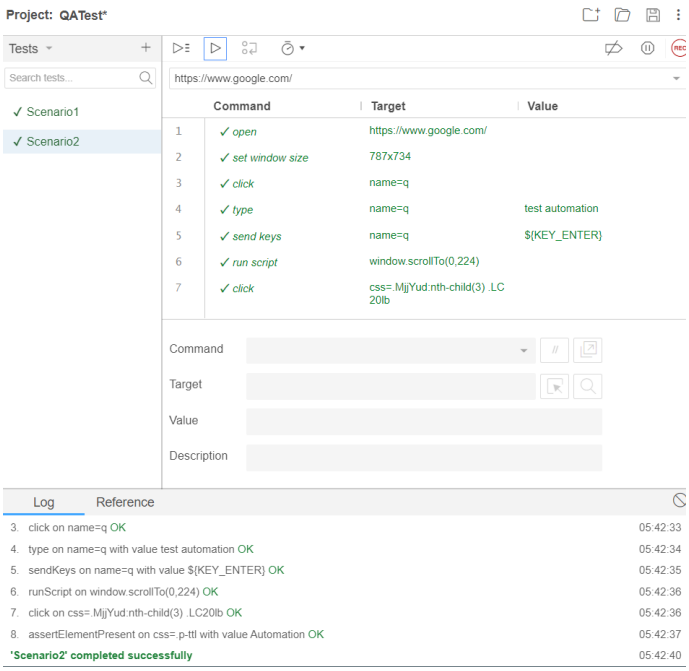


b) *Scenario 2: User can go to the first search result*

- Given I Search by keyword
- When I click on the first result link
- Then I go to the page, and the page title contains the word “automation”

Figure 2

Automation on Selenium Scenario 2



Part 2 (SQL Basic Scripting):

1. Explain the difference, in databases, between ‘Having’ and ‘where’ when it comes to a query. Develop one example for each one of this two cases and point out the difference.

   “Having” filters grouped data, while “where” does so for ungrouped data.
   Consider the table "grades" having following schema:
   - Student
   - subject
   - grade

   If you are interested on knowing which subjects where failed by at least one student, you must use the "where" statement (e.g. select distinct subject from grades where grade < 3.0). However, if you want to know which students had a GPA of over 4.0, you must use the "having" statement (select student, avg(grade) as GPA from grades group by student having avg(grade) > 4.0)

2. Write a query for create a data table ‘Student’ with the following attributes in it: ‘Name, ‘Code, ‘Class’, ‘Age’, ‘Favorite Subject, ‘GPA’ (5.0 scale).

```
CREATE TABLE IF NOT EXISTS Student
(
Name TEXT,
Code INT,
Class TEXT,
Age INT,
Favorite_Subject TEXT,
GPA float,
CHECK (GPA <= 5)
);
```

3. Insert at least 40 records in the last table with close to real data.

```
COPY students(name, code, class, age, favorite_subject, GPA)
FROM ' C:\Users\crist\Desktop\QA test'
DELIMITER ','
CSV HEADER;
```

4. Write a query to get the average of the GPA from all the students which name starts with ‘A’.

```
SELECT
   name,
AVG(gpa)
FROM
  student
WHERE
  name LIKE "A%"
GROUP BY
  name
```

5. Write a query to get the list of students that: are in the same class, have a GPA higher than 3.5/5.0 and order them by Age and Name.

```
SELECT
   class,
   name,
   age
FROM
   student
GROUP BY
   class,
  name,
   age
```

```
WHERE
   MAX(GPA) > 3.5
ORDER BY
   age,
   name
```

6. Write a query to get the list of all the students with ‘Name, ‘Code, ‘Class’, ‘Age’, ‘Favorite Subject, ‘GPA’.

```
select * from student
```

7. Take the following 25 question quiz about SQL, please include a screenshot about the results and time it took to take the test.
   http://www.w3schools.com/quiztest/quiztest.asp?qtest=SQL

Figure 3

SQL quiz, first try

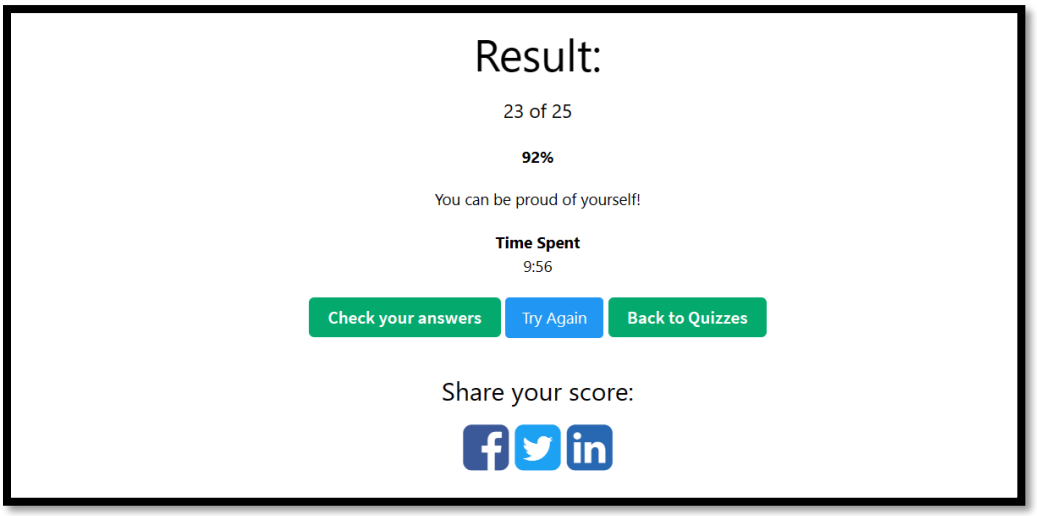
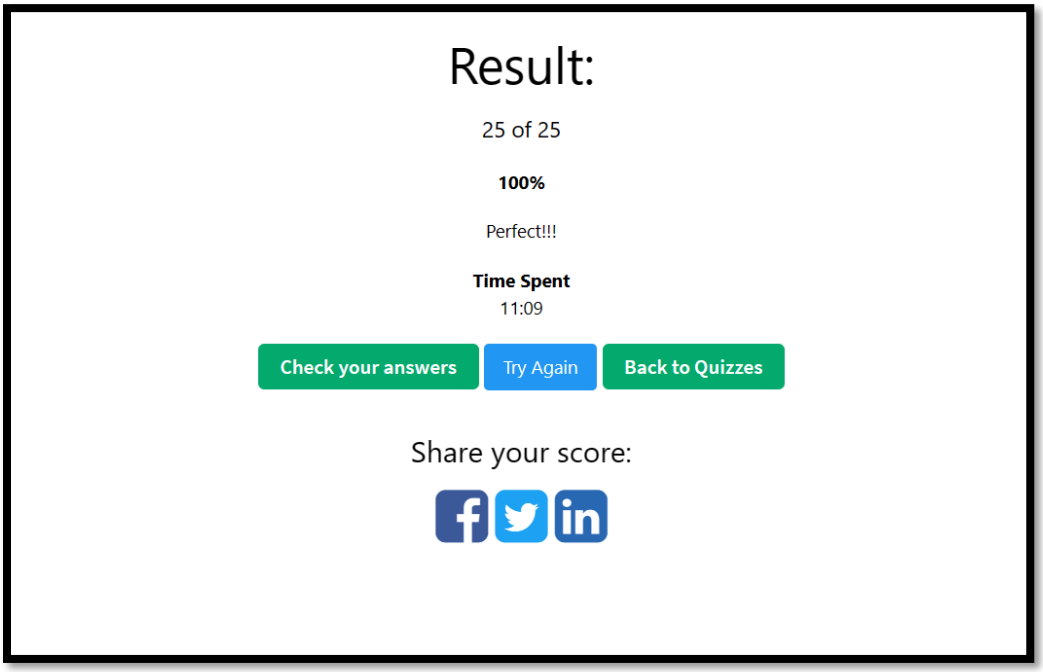


Figure 4

SQL quiz, second try



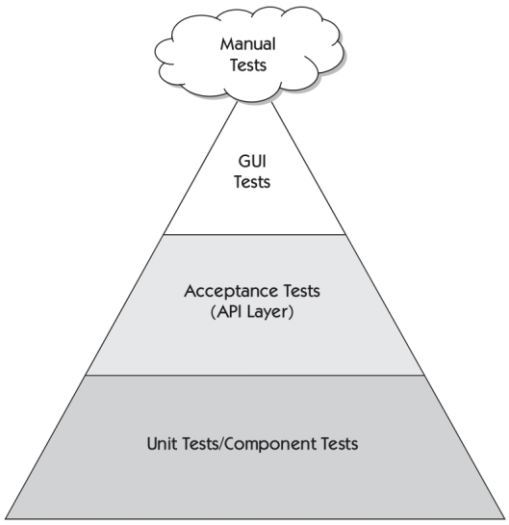
Part 3 (Software Testing Knowledge):

1. What is the difference between a unit test, an acceptance test, an integration test and an end-to-end test?

unit test	means testing individual modules of an application in isolation (without any interaction with dependencies) to confirm that the code is doing things right.
acceptance test	when a user/customer/business receive the functionality they (or your test department) will conduct Acceptance tests to ensure that the functionality meets their requirements.
integration test	means checking if different modules are working fine when combined together as a group.
end-to-end test	End-to-end testing is a Software testing methodology to test an application flow from start to end. The purpose of this testing is to simulate the real user scenario and validate the system under test and its components for integration and data integrity. It is performed from start to finish under real-world scenarios like communication of the application with hardware, network, database, and other applications.

Functional test	means testing a slice of functionality in the system (may interact with dependencies) to confirm that the code is doing the right things.

2. Could you explain Cohn's automation pyramid?



A testing pyramid is a visual representation of a testing methodology that can be applied to your QA strategy. It can help you visualize the amount of time your team should be focusing on the different testing categories or groups.

This, in turn, should help you better allocate resources and become more efficient with your QA efforts, and more effective.

The Agile Testing Pyramid is a concept developed by Mike Cohn in his book Succeeding with Agile. It helps us understand the difference between traditional testing frameworks and testing for agile development.

Even though both pyramids contain the same layers, these are placed at different levels with varying degrees of involvement. The main three layers or testing categories are:

- UI Testing
- Integration Testing
- Unit Testing

Today, most software organizations suffer from the Traditional Testing Pyramid problem, commonly known as the 'Inverted Testing Pyramid'

With an Inverted Testing Pyramid, development teams spend most of their time and effort manually checking software within an inefficient framework. Even though some try to add automation here and

there, a team can quickly fall into an 'automation paradox' - that being when the setup and maintenance of test automation take up more time and resources than what can be saved by automating.

3. Could you explain the difference between a black box testing and a white box testing?

Criteria	**Black Box Testing**	**White Box Testing**
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested *is NOT known* to the tester.	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested *is known* to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing: Acceptance Testing & System Testing	Mainly applicable to lower levels of testing: Unit Testing & Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

4. What is the purpose of an exploratory test and when is it useful to run them?

Exploratory Testing is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a "thinking" activity.

Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

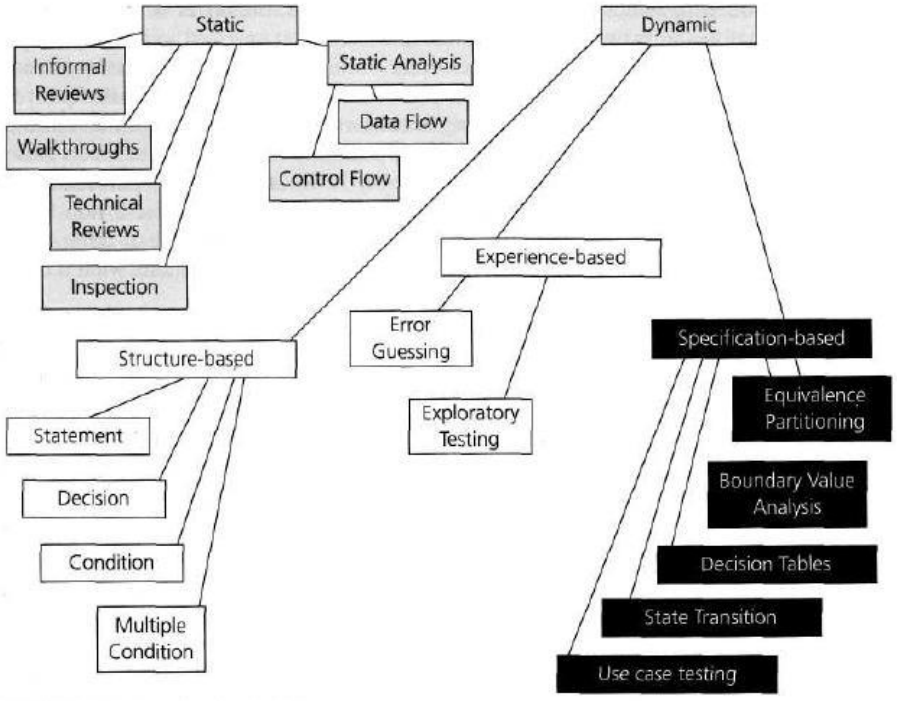
- Is not random testing but it is ad-hoc testing with a purpose of find bugs
- Is structured and rigorous
- Is cognitively (thinking) structured as compared to the procedural structure of scripted testing. This structure comes from Charter, time boxing etc.
- Is highly teachable and manageable
- It is not a technique but it is an approach. What actions you perform next is governed by what you are doing currently

5. Mention at least 5 test design techniques and explain them briefly

Each testing technique falls into one of a number of different categories. Broadly speaking there are two main categories:

Figure 5

Testing techniques. From: [What is test design technique? (tryqa.com)]



In general, Static testing is the testing of the software work products manually, or with a set of tools, but they are **not executed**. It starts early in the **software development life cycle** and so it is done during the **verification process**.

Most static testing techniques can be used to 'test' any form of document including source code, design documents and models, functional specifications and requirement specifications.

Now, let's look at Dynamic testing. The name itself suggests that it is "Dynamic" in nature, which means changing. This is the kind of testing that we do with changing values or conditions **by executing the code**.

This involves testing the application in real-time by giving inputs and examining the result or the output value of behavior.

a) Boundary Value Analysis (BVA)

Boundary value analysis is based on testing at the boundaries between partitions. It includes maximum, minimum, inside or outside boundaries, typical values and error values.

It is generally seen that a large number of errors occur at the boundaries of the defined input values rather than the center. It is also known as BVA and gives a selection of test cases which exercise bounding values.

This black box testing technique complements equivalence partitioning. This software testing technique base on the principle that, if a system works well for these particular values then it will work perfectly well for all values which comes between the two boundary values.

b) Equivalence Class Partitioning

Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same. This software testing method divides the input domain of a program into classes of data from which test cases should be designed.

The concept behind this Test Case Design Technique is that test case of a representative value of each class is equal to a test of any other value of the same class. It allows you to Identify valid as well as invalid equivalence classes.

c) Decision Table Based Testing

A decision table is also known as to Cause-Effect table. This software testing technique is used for functions which respond to a combination of inputs or events. For example, a submit button should be enabled if the user has entered all required fields.

The first task is to identify functionalities where the output depends on a combination of inputs. If there are large input set of combinations, then divide it into smaller subsets which are helpful for managing a decision table.

For every function, you need to create a table and list down all types of combinations of inputs and its respective outputs. This helps to identify a condition that is overlooked by the tester.

d) State Transition

In State Transition technique changes in input conditions change the state of the Application Under Test (AUT). This testing technique allows the tester to test the behavior of an AUT. The tester can perform this action by entering various input conditions in a sequence. In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior

e) Error Guessing

Error Guessing is a software testing technique based on guessing the error which can prevail in the code. The technique is heavily based on the experience where the test analysts use their experience to guess the problematic part of the testing application. Hence, the test analysts must be skilled and experienced for better error guessing.
The technique counts a list of possible errors or error-prone situations. Then tester writes a test case to expose those errors. To design test cases based on this software testing technique, the analyst can use the past experiences to identify the conditions.

6. What is the purpose of the following types of tests?

a. *Functional test:* confirm that the code is doing the right things
b. *Performance test*: evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload. Organizations will run performance tests to identify performance-related bottlenecks.
c. *Security test*: is a type of software testing that intends to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders. It falls under non-functional testing.
d. *Usability test*: is a method used to evaluate how easy a website is to use. The tests take place with real users to measure how 'usable' or 'intuitive' a website is and how easy it is for users to reach their goals.
e. *API test*: involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security. API testing is now considered critical for automating testing because APIs now serve as the primary interface to application logic and because it can greatly improve the efficiency of your testing strategy as a whole, helping you deliver software faster than ever.
f. *Unit Test*: the porpose is to test individual modules of an application in isolation (without any interaction with dependencies) to confirm that the code is doing things right.

References:

- https://www.softwaretestinghelp.com/the-difference-between-unit-integration-and-functional-testing/
- https://stackoverflow.com/questions/7672511/unit-test-integration-test-regression-test-acceptance-test
- https://www.softwaretestinghelp.com/what-is-end-to-end-testing/
- https://www.testingexcellence.com/exploratory-testing-important-agile-projects/
- https://www.guru99.com/exploratory-testing.html
- http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/
- https://smartbear.com/solutions/api-testing/
- http://softwaretestingfundamentals.com/security-testing/
- https://www.guru99.com/what-is-security-testing.html
- https://www.experienceux.co.uk/faqs/what-is-usability-testing/
- https://en.wikipedia.org/wiki/API_testing
- https://en.wikipedia.org/wiki/Application_programming_interface
- https://searchsoftwarequality.techtarget.com/definition/performance-testing
- https://www.guru99.com/performance-testing.html
- https://www.tutorialselenium.com/2017/05/28/como-usar-selenium-ide/
- https://cleventy.com/tutorial-selenium-primeros-pasos/
- http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/381
- https://www.tutorialselenium.com/2017/09/24/como-usar-comandos-de-selenium-ide/
- https://testeandosoftware.com/selenium-comandos-selenese/
- https://www.seleniumhq.org/selenium-ide/docs/en/introduction/getting-started/
- http://www.cs.tau.ac.il/~amiramy/SoftwareSeminar/CTDmay2012.PDF
- https://www.guru99.com/decision-table-testing.html
- https://www.toolsqa.com/software-testing/decision-table-testing/
- https://www.tutorialspoint.com/software_testing_dictionary/data_flow_testing.htm
- https://www.javatpoint.com/data-flow-testing-in-white-box-testing
- https://www.thedigitalmentor.com/what-is-dataflow-testing/
- https://en.wikipedia.org/wiki/Boundary-value_analysis
- https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html
- https://www.testingexcellence.com/boundary-value-analysis/
- http://www.professionalqa.com/combinatorial-testing
- https://en.wikipedia.org/wiki/Classification_Tree_Method
- https://www.expleo-germany.com/en/products/testona/classification-tree-method/
- https://inf.mit.bme.hu/sites/default/files/materials/taxonomy/term/445/13/04_Testing.pdf
- https://www.bcs.org/upload/pdf/amettehass-131211b.pdf
- https://www.bcs.org/upload/pdf/amettehass-131211a.pdf

- https://www.tutorialspoint.com/software_testing_dictionary/pdf/test_case_design_technique.pdf
- https://testautomationresources.com/software-testing-basics/software-test-design-techniques/
- https://www.uio.no/studier/emner/matnat/ifi/INF3121/v18/forelesningsvideoer/chapte-4-part-2-slides.pdf
- http://tryqa.com/what-is-test-design-technique/
- http://tryqa.com/what-is-structure-based-technique-in-software-testing/
- http://tryqa.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/
- http://tryqa.com/what-is-white-box-or-structure-based-or-structural-testing-techniques/