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

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.
  1. 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”
  2. 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”

Guidelines:

You are testing https://www.google.com.

Always make use good principles and practices when designing your Solution.

Implement your automation solution, if possible, following the Page Object Model pattern and BDD paradigm.

Solution for part 1 at <https://github.com/herrmonkey/QA-Test>

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.

The difference between the 'Having' and 'Where' clauses in a database query is that the 'Where' clause is used to filter data from a result set based on certain conditions, while the 'Having' clause is used to filter data from a result set based on aggregate functions.

For example, if we wanted to retrieve all the employees from a database who work at a certain department and have a salary greater than $50,000, we could use the following query using the 'Where' clause:

SELECT \* FROM employees WHERE department = 'Marketing' AND salary > 50000;

Using the 'Having' clause, we could use the following query instead:

SELECT department, AVG(salary) FROM employees GROUP BY department HAVING AVG(salary) > 50000;

The difference here is that the 'Where' clause is used to filter the rows of the table based on certain conditions, while the 'Having' clause is used to filter data from the result set based on aggregate functions such as AVG().

1. 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 Student ( Name VARCHAR(255), Code VARCHAR(10), Class VARCHAR(255), Age INTEGER, Favorite\_Subject VARCHAR(255), GPA DECIMAL(3, 1) );

CREATE TABLE Student (

Name VARCHAR(255),

Code INT,

Class VARCHAR(255),

Age INT,

Favorite\_Subject VARCHAR(255),

GPA DECIMAL(2,1)

);

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

INSERT INTO Student(Name, Code, Class, Age, Favorite\_Subject, GPA)

VALUES (1value1, 1value2, 1value3, 1value4, 1value5, 1value6), (2v1, 2v2, 2v3, 2v4, 2v5, 2v6), …(40v1, 40v2, 40v3, 40v4, 40v5, 40v6);

INSERT INTO Student (Name, Code, Class, Age, Favorite\_Subject, GPA)

VALUES('Ricardo', 70330247, 'Computer Science', 37, 'Innovation', 4.0),

('Oscar', 70329732, 'Computer Science', 39, 'Math', 5.0), …, ('Andres', 1034234785, 'Computer Science', 30, ''Computer Science ', 5.0);

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

SELECT AVG(GPA)

FROM Student

WHERE Name LIKE 'A%';

1. 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.

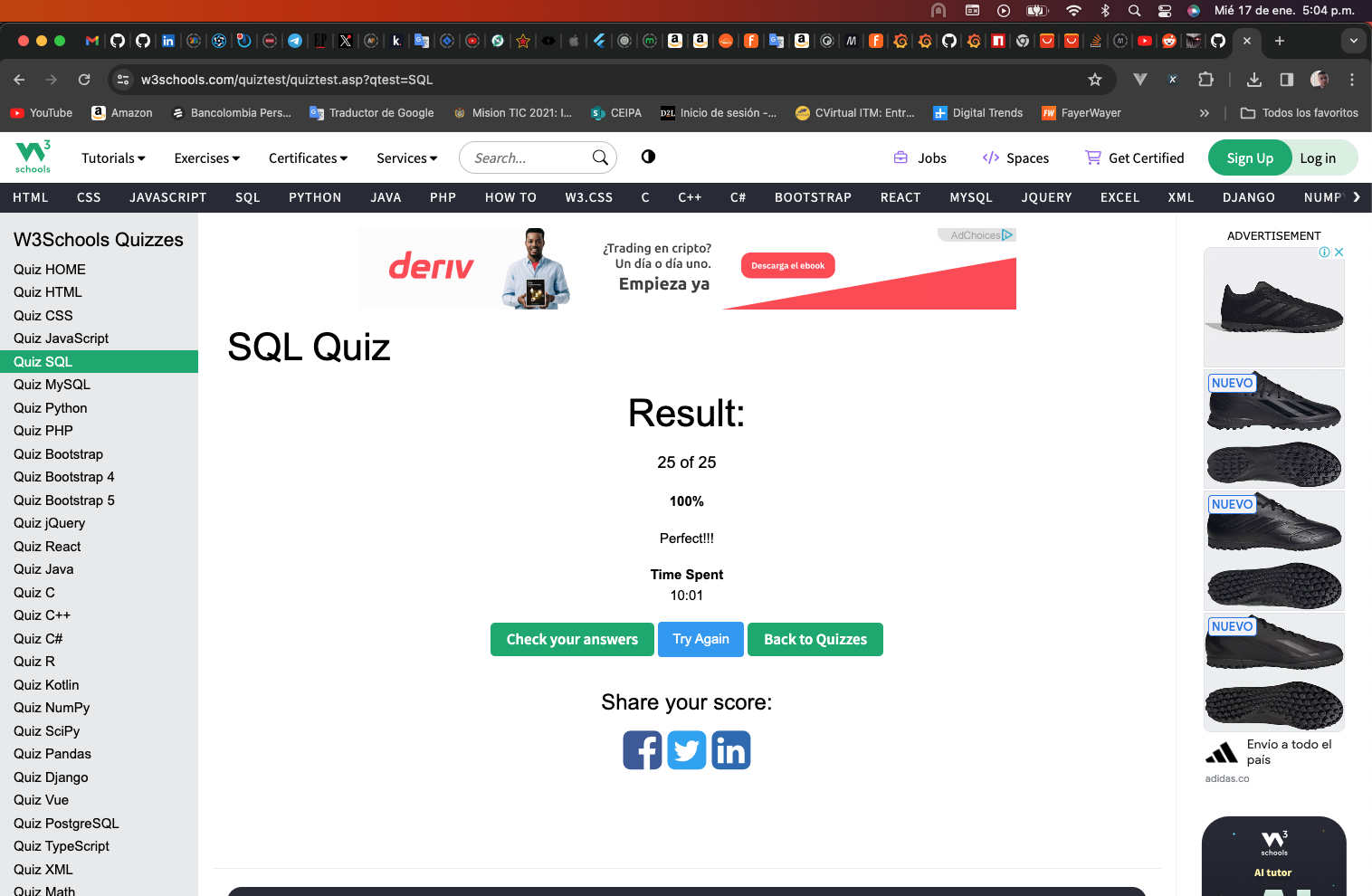
SELECT \* FROM Student WHERE Class = 'Computer Science' AND GPA > 3.5 ORDER BY Age, Name;

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

SELECT Name, Code, Class, Age, Favorite\_Subject, GPA FROM Student;

1. 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>



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?

The main differences between unit, acceptance, integration, and end-to-end tests are in the scope and purpose of the tests. Unit tests are the most granular and aim to test the smallest possible units of code, such as individual functions. Acceptance tests are used to verify that the system meets the requirements specified by the customer. Integration tests are used to check the integration between the components of the system, while end-to-end tests are used to ensure that the entire system works as intended.

1. Could you explain Cohn's automation pyramid?



Mike Cohn's automation pyramid is a way of conceptualizing different levels of testing, with unit tests forming the base and end-to-end tests forming the peak. Unit tests are the most granular and focus on testing the smallest possible units of code, such as individual functions. Integration tests are used to check the integration between the components of the system, while acceptance tests are used to verify that the system meets the requirements specified by the customer. End-to-end tests are the highest level and are used to ensure that the entire system works as intended. The idea behind the pyramid is that you should focus on writing more unit tests and fewer higher-level tests, since unit tests are the quickest and most reliable way to test code.

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

Black box testing is a type of software testing that focuses on the functionality of an application without looking at the internal structure or code. It is used to test the system's external behavior and the usability of the application. On the other hand, white box testing looks at the internal structure and code of the application to test its functionalities. It is used to test the application's code and implementation, and requires knowledge of the implementation. White box testing is often used to test boundary cases and find any errors or bugs in the code.

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

Exploratory testing is a type of black-box testing that involves the tester exploring the system to gain a better understanding of its features and functionality. It is useful to run exploratory tests when you want to identify potential issues that may not be obvious or covered by existing tests. Exploratory testing is also useful when you need to quickly test a feature or investigate a bug, as it allows the tester to quickly analyze the system and get to the root of the issue.

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

There are several test design techniques that can be used to create effective tests for software products. The five most common techniques are: Equivalence Class Partitioning, Boundary Value Analysis, Error Guessing, Cause-Effect Graphing, and Use Case Testing.

Equivalence Class Partitioning: This technique partitions the input and output domains into equivalence classes, which are groups of values that can be considered the same for the purpose of testing. This technique helps to identify test cases for valid, invalid, and boundary conditions.

Boundary Value Analysis: This technique identifies test cases at the boundaries of the input and output domains, which can help identify any unexpected behavior at the boundaries.

Error Guessing: This technique involves making educated guesses about potential errors or defects in the software, which can then be tested to determine if they are correct.

Cause-Effect Graphing: This technique involves mapping out potential causes and effects of errors or defects, which can help in identifying and testing potential problems that may not be immediately apparent.

Use Case Testing: This technique involves testing the use cases that the software is designed to support, which can help to identify any problems that may arise when users interact with the software in the way it was intended to be used.

1. What is the purpose of the following types of tests?
   1. Functional test:

The purpose of a functional test is to validate that a system or component is working as expected, by testing the functionality of the system against the requirements. Functional tests are typically divided into unit tests, integration tests, and end-to-end tests. Unit tests are used to test individual components in isolation, integration tests are used to test the interactions between components, and end-to-end tests are used to test the entire system from start to finish. Functional testing is an important part of the software development process and can help to identify any errors or defects in the software before they are released to the public.

* 1. Performance test:

The purpose of a performance test is to measure the performance of a system or application under a given workload. Performance tests can be used to measure the speed, scalability, reliability, and responsiveness of a system, as well as to identify potential bottlenecks or areas for improvement. Performance tests are usually conducted in a controlled environment, with specific input and output parameters, to accurately measure and compare the performance of different systems. Performance tests can also be used to identify potential errors or defects in the system before they are released to the public.

* 1. Security test:

The purpose of a security test is to identify any potential vulnerabilities or security risks in a system or application. Security tests are typically conducted by security professionals using manual or automated techniques, such as penetration testing, vulnerability scanning, and code review. The security tests can help to identify any potential weaknesses that could be exploited by malicious actors, allowing organizations to address these issues before they can be exploited.

* 1. Usability test:

The purpose of a usability test is to evaluate how easy it is for users to use a product or service. Usability tests typically involve observing users as they interact with the product or service and recording their feedback. The tests are designed to identify any problems or issues that users may experience while using the product or service, and to identify any areas for improvement. Usability tests can help to ensure that products and services are easy to use and intuitive for users, resulting in a better user experience.

* 1. API test:

The purpose of an API test is to validate that an application's application programming interface (API) is functioning correctly. API tests involve sending requests to the API and validating the responses. These tests are typically automated and can be used to simulate various types of user interactions with the API, such as sending requests with different parameters and validating the responses. API tests can help to identify any errors or defects in the API before they are released to the public.

* 1. Unit Test:

The purpose of a unit test is to validate that individual components of a system or application are working as expected. Unit tests are typically written by developers and are used to test individual functions or classes in isolation. Unit tests can be used to verify that the code is functioning as intended, identify any errors or defects in the code, and ensure that new changes or additions to the code do not introduce any new errors. Unit tests can also be used to verify the correctness of the code, and to ensure that the code is following best practices.

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/>