Technical Test – QA automation

Exercise 1: API validation, data retrieval, and calculations.

Description:

This exercise requires the use of a free API to retrieve dynamic information in JSON format. You must validate multiple properties, perform calculations using the retrieved data, and ensure the integrity of the information.

Requirements:

Use Java (preferably) for the following exercise:

- Consume the API: https://dummyjson.com/products
- For each request made:
- Validate that the response status code is 200.
- Print the response time of the endpoint.
- Create test cases that cover the validation requirements.

Development:

- Extract six products under the following conditions:
    - Product categories must not be null.
    - Retrieve three products from the "groceries" category.
    - None of these products should have a price greater than $5.
    - Retrieve three products from the "beauty" category.
    - None of these products should have a price lower than $5 or higher than $14.

Expected Outcome:

- Using the information from the six retrieved products, perform the following calculations:
    - Sum the price of all products to obtain a total price.
    - Calculate the overall average price of the six products.
    - Calculate the average price per category (groceries and beauty).

- Display the following data in the Console for each processed product:
    - Id
    - Title
    - Category
    - Price

Exercise 2: Full purchase flow in an online store

Description:

This exercise involves automating a complete purchase flow on a test website. Multiple steps must be performed, ensuring that products are correctly added to the cart and verifying tax and total calculations.

Requirements:

Use Selenium in Java.

Development:

- Access https://www.saucedemo.com/
- Log in with the following credentials:
    - Username: standard_user
    - Password: secret_sauce
- Add four products to the cart and verify the subtotals.
- Proceed to checkout and fill in the shipping details.
- Print the following values obtained from the web process in the console:
    - Item total
    - Tax
    - Total
- Complete the purchase.

Expected Outcome:

- Complete the purchase and verify the success message.

Exercise 3: Mobile test automation - Wikipedia App

Description:

This exercise assesses the ability to automate interactions within a mobile application, focusing on dynamic elements, content validation, and navigation flows.

Requirements:

- Use Appium with Java.
- Utilize an Android Emulator (via Android Studio) or a physical Android device.
- Download and install the "Wikipedia" app from the Google Play Store (The application may already be installed on the device beforehand; it is not necessary to install it during execution).

Development:

- Launch the Wikipedia app and verify that it loads successfully.
- Navigate to the search section.
- Enter the search term "Selenium" and initiate the search.
- Select the first result from the search results.
- Scroll to the "History" section.
- Return to the home screen of the app.

Expected Output:

- The application launches without issues.
- The search function works correctly, displaying relevant results.
- The selected article contains a "History" section with appropriate content.
- The test executes all interactions without failures, logging results to the console.

Important:

- All tests must be implemented within the same solution.
- Each test execution result must be reflected in a report (reporting tool of your choice).

Optional but recommended:

- Include test evidence in the report (screenshots for web executions).
- Include both request and response details in the report.

Evaluation Criteria:

- Review of the test source code and its architecture.
- File naming conventions and consistency.
- Proficiency in the programming language and/or testing framework used.
- Structure and organization of the solution.
- Documentation of the solution and test cases.
- Gitflow adherence and version control practices.
- Best practices for code reusability and maintainability.
- If any of the three tests are not fully developed within the stipulated time, the logic behind the implementation and the approach taken to solve the problem will be considered.

Submission:

- A **public repository link** containing the solution along with its documentation (Include **README.md**).

Deadline:

- 3 days from the date and time the challenge is sent.