Test Task: Junior QA Automation Engineer

Task 1:

Write an End-to-End Test

You can use any language or library. Please use the desktop (> 1000px width) version of this page.

1. Check if the page title exists
2. Check if the grade is visible and is above zero (e.g. 4.81/5.00)
3. Check if the “Wie berechnet sich die Note?” link opens the window with additional information and check if the provided information is relevant
4. Click on "2 Stars" to filter all "two stars" reviews - ensure that every review in the entire list has only two stars
5. Create the sum of all star percentage values. The sum must be equal or below 100.

Web automation task:

```
# Library:
''' The list below contains the library files imported from
Selenium tool '''

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
EC


# Driver Initialisation:
''' The below code initiates driver to the corresponding
browser by its executable file '''

chrome_path = "C:\\chromedriver.exe"
service = Service(chrome_path)
driver = webdriver.Chrome(service=service)
wait = WebDriverWait(driver, 10)


# Maximize Window:
''' The below code maximizes the browser window for better
screen resolution and experience '''

driver.maximize_window()


# Launch URL and Title verification:
''' The below code verify the page title exists or not '''
```

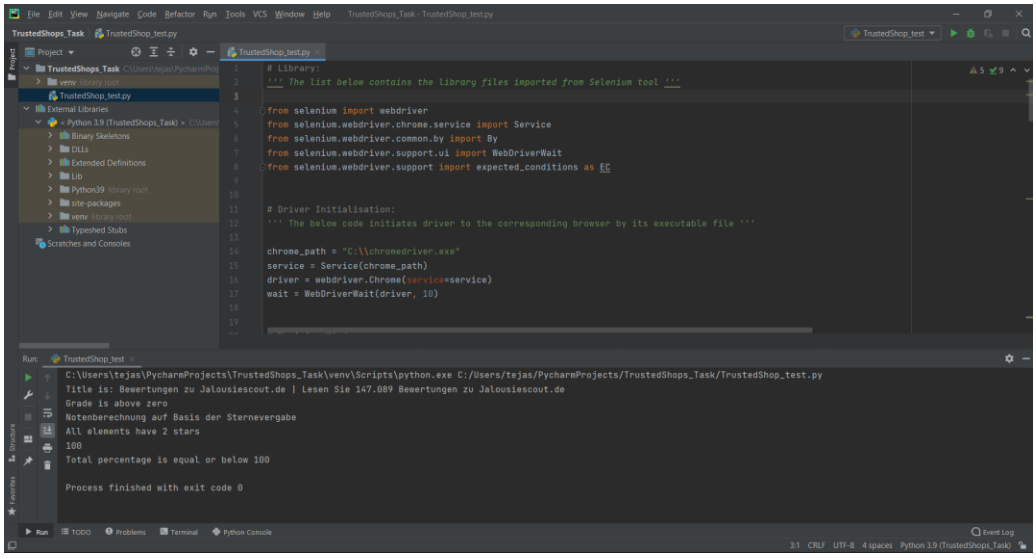
- by, Tejaswini Arukati

```
driver.get('https://www.trustedshops.de/bewertung/info_X77B11C
1B8A5ABA16DDEC0C30E7996C21.html')
if driver.title != " ":
    print("Title is:", driver.title)
else:
    print("no title")
driver.implicitly_wait(5)


# Grade verification and validation:
''' The below code verify the grade is visible and validates
it is above zero '''

element = wait.until(
    EC.visibility_of_element_located((By.XPATH,
"//*[@id='top']/div/div[4]/div[2]/div[1]/div[1]/div[2]/span"))
).text
grade_value = float(element.replace(',', '.'))
if grade_value > 0:
    print('Grade is above zero')
else:
    print('Grade is zero or below')



# Click On Link and the Information is relevant:
''' The below code verify the link is clickable and the link
window Information is relevant or not '''

driver.find_element(By.LINK_TEXT,"Wie berechnet sich die
Note?").click()
newwindow = driver.find_element(By.XPATH, "//pre[normalize-
space()='Notenberechnung auf Basis der Sternevergabe']").text
print(newwindow)
assert "berech" in newwindow



# Click on "2 stars" to filter all two star reviews and ensure
the results are relevant :
''' The below code verify the "2 stars" are clickable which
filters all two star reviews and ensuring every review in the
list has only two star rating '''

link_element =
wait.until(EC.element_to_be_clickable((By.XPATH,
"//a[@href='/bewertung/info_X77B11C1B8A5ABA16DDEC0C30E7996C21.
html?stars=2']")))
driver.execute_script("arguments[0].click();", link_element)
review_elements = driver.find_elements(By.XPATH,
```

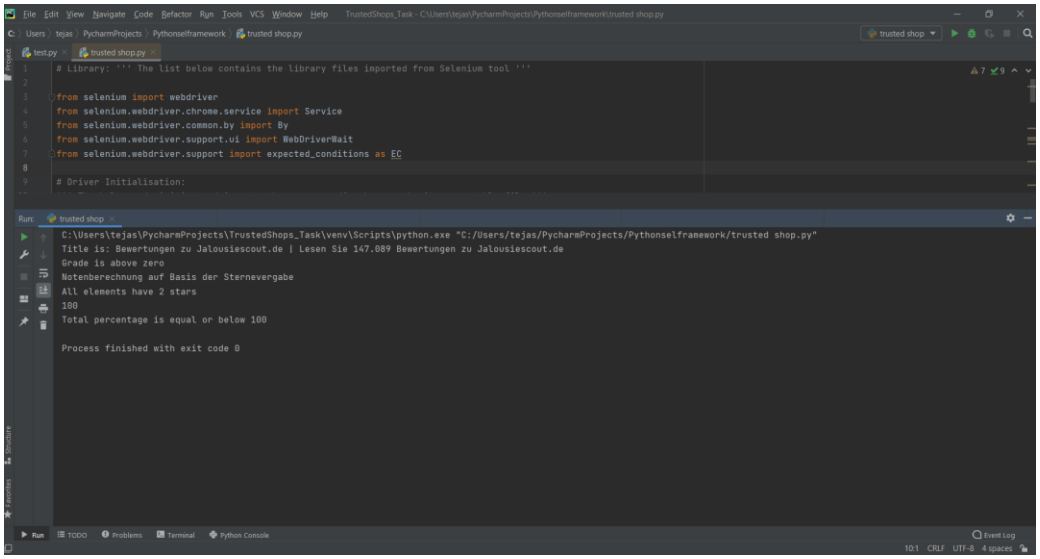
```
"//div[@class='sc-2e7612c5-0 sc-f836bc46-0 kyZgbN chcERM']"

"//div[@class='Starsstyles__Stars-sc-4o1xbr-0 gWZgUz']")
count_2_stars_for_all = True  # Flag to track if count is 2
for all elements

for element in review_elements:
    span_elements = element.find_elements(By.XPATH, ".//span")
    count = len([span for span in span_elements if "color:
rgb(255, 220, 15)" in span.get_attribute("style")])   # To
count only two stars using color style for the entire star
rating component
    if count != 2:
        print("The count is not 2 for this element")
        count_2_stars_for_all = False
        break  # Exit the loop if any element has a count
different from 2
if count_2_stars_for_all:
    print("All elements have 2 stars")



# Sum of all star percentage values must be equal or below
100:
''' The below code verify the sum of all star percentage
values must be equal or below 100 '''

star_elements = driver.find_elements(By.XPATH,
"//div[@class='sc-61f2e426-3 sc-61f2e426-4 lmrSdC ghcBqu']")
total_percentage = 0
for star_element in star_elements:
    total_percentage +=
round(float(star_element.get_attribute("style").split(":")[1].
strip("%;")))   # To round off the percentage values of each
star rating and stores as Int value

print(total_percentage)

if total_percentage <= 100:
    print("Total percentage is equal or below 100")
else:
    print("Total percentage is above 100")
driver.quit()




''' End of code '''
```

PyCharm structure view:

- by, Tejaswini Arukati



Task Output:



Concept for further development:

1. What kind of checks would you add to the current tests?

   I would add the assertion and verification checks for the current tests to validate the contents are responding to the expected content.

2. What are the next tests you would recommend implementing?

   Firstly, I would like to implement the most critical features of the following:

   - Add a test to verify the trusted shops header with logo, search bar, Information drop-down and login on it and their functionality.
   - Add a test to verify a company profile is displayed along with company logo.
   - Add a test to the “verified” checkmark for the company profile

- by, Tejaswini Arukati

- Add a test to the reviews count under the company showing up or not.
- Add a test to verify the review count matches under the company profile name and the total reviews count under the star ratings component.
- Add a test whether most relevant positive and critical experiences show up for the company profile.
- Add a test to verify the 'All positive reviews' and 'All critical reviews' are clickable and functionable.
- Add a test to the "Verifizierte Bewertung" (verified rating) for each review posted by the user for all star ratings whether verified or not to check the authenticity.
- Add a test for the answers for the reviews whether if there are any unanswered customer reviews.
- Also, I will add a test for the '**Gütesiegel' (**seal of approval) section whether the seal is valid or not and their last updated date.
- Add a test for the search functionality for the reviews search, filter, and sorting.
- Add a test to verify the helpful button is clickable and functional.
- Add a test to verify the report review button is clickable and functional.
- Add a test to verify the required data about the corresponding company is relevant or not in 'Details on 'ABC' company'.
- Add a test to verify the 'Trusted Shops quality criteria' link works and shows relevant data.
- Add a test to verify the pagination works for the reviews results.

3. Does the profile page have non-functional properties? How would you test these properties?

   Yes, the profile page has non-functional properties such as:

   - Reviews under the company profile name- would test by verifying with the total review count under the star ratings component.
   - Star colors are displayed properly.
   - Color status on top of most positive and most critical experiences and their rating colors are displayed properly.
   - The "verified" checkmark for the company profile, reviews, most relevant positive and critical experiences.
   - Load testing on the reviews for all the star ratings.

Task 2:

- by, Tejaswini Arukati

A) Describe how you would approach the discovered issues, develop a plan for the team to improve on the findings.

As a QA I would like to make sure the process works out after each release is to have a Release management checklist.

1. **Plan your release**:
   The first step in a successful release management process is, unsurprisingly, to plan the release. The details of how your organization approaches release planning depend on the specifics of your software development process. For instance, agile teams will plan releases according to approved user stories. The stakeholder should be leading this activity to check what and all to be planned for release.

2. **Build and Test the release:**
   Building and testing should happen for the features intended to go in the release. Building and Testing should happen in parallel with the help of TDD or BDD approach. Early testing will ensure more bugs in earlier stage of software that will result in reduction of production bugs.
   - *This approach would minimize the creation of bugs after release and could avoid the situation of disabling the tests and delaying the release process by fixing bugs*.
   - *Also, the TDD approach would save ample amount of time for both devs & QAs, to identify the bugs earlier in the development phase which could result in not missing the sprint goal.*

3. **Perform UAT Testing.**
   User acceptance testing (UAT) consists of a process of verifying that a solution works for the user. This form of testing consists of people from the target audience (one can be a product owner) using the application to determine whether it is easy to use and understand and whether it meets its requirements.

4. **Regression testing:**
   Perform regression testing on the new feature along with the old features before each release to verify that it still works as expected. This should include testing all the UI elements and features, Use automated testing tools to test the system and ensure that it meets all the requirements.
   - *This testing helps the product to get the results of feature working conditions whenever any developer touches any piece of code at any time. If any new scenarios found, then can be added in the test scenarios list in the test plan and then automate it accordingly.*

5. **Load testing:**
   Conduct load testing on the system to ensure that it can handle the expected amount of traffic. This will help identify any performance issues that may arise due to increased usage.
   - *Regular performing of this testing would find out the root cause of system slowness and get ready for the worst a way ahead before experiencing by the user.*

- by, Tejaswini Arukati

6. **Security testing:**
   Test the security measures implemented in the system to ensure that they are still effective in protecting Intruders from unauthorized access.

7. **Continuous improvement:**
   Continuously monitor the features and implement improvements to ensure that it meets the changing needs of users. Regularly review the system to identify any potential issues and address them promptly. If any bugs identified should be immediately informed to the team by considering the severity and priority of the bug. Employ manual testing only when it is strictly necessary. Automate anything else to achieve high test coverage.
   - *Yes, to maintain or improve the code quality we, the product team must dedicate some tickets in every sprint to fix them and give the possible best features to users and by maintaining this time to time gradually can increase the quality product over the time.*

   By taking these steps, it is possible to ensure that the process continues to work properly after each release, providing users with a smooth and reliable experience.

B) Create a summary for your team in order to convince them to support your plans.

As a QA, I consider following below practices and would convince the team, also by taking suggestions for any improvements in achieving best outcomes in a delivery process.

• **Understand the Requirements thoroughly before starting the QA procedures** - Any confusion in any of the statements or concept should be cleared immediately, in the requirement analysis phase itself, from the product owner or business analyst.Grooming sessions can be organised with the whole team to discuss the feature.

• **Prepare a detailed Test plan** - A well detailed Test plan comprising of Feature to be tested, scope of testing, features not to be tested, Environment of testing, Test data required for testing, any specific tools to be used for testing, its version, resources utilised, estimation and efforts with completion date, risks etc.

- **Prepare testcases in detail in a clear and simple manner** - Write the test cases in simple English where each step should define the action clearly. Maximum test coverage should be achieved.
- **Effective communication within team** - Any issues during testing should be communicated to the team . The bugs identified should be immediately informed to the team and involve in prioritising the bug fixing.
- **Create bugs with all the necessary details**- The bug should be created with at least the following informations like Summary , steps to reproduce the bug, expected result , actual result , used browser, test data , screenshots or video recording,

priority , severity ,assignee and comments if any. This would save the time of developer coming back to QA for the bug details.

- **Effective utilisation of project management tool** - The QA should have a ticket assigned for the current task. Each tickets (task,testcase,bugs) should be closed with proper comments. Each tickets should be linked properly to its parent ticket.
- **Maintain a regression suite** - Create a manual regression suite with necessary functional tests. If the project goes live regularly, then this suite can be updated before each release. Once the functional testing is completed, this suite should be updated with the major test cases of the features going live for coming release. During the regression phase of the release, this suite alone can be tested as Regression . This would cover all the previous and current features thereby ensuring nothing breaks. The same manual regression suite can be taken as a reference and implement the automation scripts which can be utilised further for regression thereby reducing manual efforts.

- by, Tejaswini Arukati