Nagy Dávid Martin

Használtauto.hu homework task cover letter

# Task

1) Látogass el a https://www.hasznaltauto.hu/regisztracio oldala!

a) Írj automata tesztet 3 általad választott különböző mezőre, ami a mezőhöz

kapcsolódó hibaüzenet megjelenését vizsgálja!

b) Készíts automata tesztet, ami a sikeres regisztrációs folyamatot vizsgálja!

i) Milyen nehézség(ek)be ütköztél?

ii) Van e ötleted a probléma megoldására?

2) Látogass el a következő oldalra:

https://www.hasznaltauto.hu/szemelyauto/nissan/qashqai/nissan\_qashqai\_1.3\_dig-t\_

mild\_hybrid\_tekna\_x-tronic\_fix\_0thm\_tovabbi\_szinekben\_is\_elerheto-18813851#sid=

e941e2f9-d2f4-49cb-b0f2-aac3a5e421f6!

Ha a hirdetés nem elérhető, akkor válassz egy neked tetsző hirdetést az oldalon a

következő feladatok elvégzéséhez:

a) Sorolj fel 5 olyan funkciót az oldalon, amit szerinted érdemes lenne tesztelni!

b) Írja automata tesztet, ami hozzáadja az adott hirdetést a parkolóhoz!

c) Válassz ki legalább kettőt a fenti funkciók közül és írj rájuk automata tesztet!

# About the solution

# Implementation

I have solved the above task in two standalone projects, one using WebdriverIO, and one with Playwright.

Both of the projects

* use POM (Page Object Model) architecture
* DDT (Data Driven Testing) where the software and the testcase made it viable
* have environment variables set up and used for common (environment dependent) variables

Both of the projects use the same testing logic. and test case separation / segmentation. This is to have one variable fixed (the testing logic) and to put the focus to the frameworks.

The USER\_EMAIL and USER\_PASSWORD variables found in .env are the credentials for the user I created as testing data for the 2/A-B task. (The email is generated via 10 minute email, thas is why it looks a bit suspicious)

## Versions

* Node: v18.9.1
* Npm: 10.2.4
* Operating system: Windows 11
* CLI: (Git) Bash
* IDE: Visual Studio
* Browser: Google Chrome 126.0
* Java (for Allure report generation): "21.0.2" 2024-01-16 LTS

The commands found in the project readmes are executed in Git Bash CLI, in the IDE terminal.

## Webdriver IO

I have made most of solution my career with this framework, and I wanted to demonstrate my ability how I creatre such solutions with a framework I already know.

After installing node modules the tests can be started with the following commands:

* npm run test-full // Runs both test spec files
* npm run test-listing // Runs the Listing page spec file
* npm run test-register // Runs the registration page spec

## Playwright

After I have created the project in WebdriverIO, I wanted to demonstrate the ability, that I can pick up a new framework. Altough I have no professional experience in this framework, Playwright was (is) easy to pick up and develop in it. The documentation is clear, and the features are logically built up. It has a lot of similarities with WebdriverIO, made a bit easier.

After installing node modules the tests can be started with the following commands:

* NODE\_ENV=development npx playwright test // Runs both test spec files
* NODE\_ENV=development npx playwright test tests/listing-details-and-parking-lot.spec.ts // Runs the Listing page spec file
* NODE\_ENV=development npx playwright test tests/registration.spec.ts // Runs the registration page spec

Providing the --ui flag after the command will open it in UI mode, where the playwright features are visible

# The Answers

## 1/A

Please check the source code for the implementation. – registration.spec.ts

This part of the solution uses DDT, where I set up the the testing date for the following logic: Given one field on the registration page, prompted with an input text and expected an error field text. Created 6 cases across 3 fields loops through these scenarios and check agains the expectation declared in the test case.

The following error field have been checked:

* Name field
* Email field
* Password field

I have created these test cases without validating all the of the possible scenarios (error texts which might occure), These are just to demonstrate the logic behind it, open for extension.

## 1/B

Please check the source code for the implementation. – registration.spec.ts

The Prod solution prevented me from making a successful registration with the automation (so the robot checks works as expected). However I created a script, which fills out the required fields with valid data, to go with to process, and I have commented the expectations and possible solutions in to the project itself. The below text will be a copy of what can be found int he solution as comments. These lines also answer the question 1/B/i and 1/B/ii

Difficulties

1) The I am not a robot check

- This prevents the happy path of the registration

- This could be solved with further investigation in the topic, my ideas are:

- according to the link above, this checks the mouse movement and behaviour of the user, so

creating something user like, it is possible that this could be bypassed

- Do it in some other technology. I have been able to bypass such scurity measures in my own projects

with python by manipulating the standalone browser and mouse movement (Because now when I run the script it

is prompted that "Chrome is being controlled by automated software"), but if the standalone python script starts

a standalone browser, the browser is not able to differentiate it from a real usecase

- The simplest: Leave this feature out of the test and dev environment, and test it manually

2) PostalCodeOption element

- PostalCodeOption DOM element disappears upon losing focus, this made locating the element pretty uncomfortable.

I was able to locate it and solve the problem, but from automation efficiency this could be improved from SW side

3) Regardless of the fact that I was not able to create a successful registration, the SW expects the user to verify his/her

email address via the confirm email. This is also not trivial solution, but there are a couple of ways I should be able to bypass this

- 1) Using 10 minute email service from where I am able

- Either fetch it through API (did not dig that through, but )

- Either load up the page and click on the link

- 2) Using some mail provider which we have the API for

- 3) Knowing the business logic behind uuid generation in the confirm link / fetching it from API and opening a new tab and searching for

the assembled link

- 4) Based on product expectations the test case could even end here and check the fact that the email is sent out,

and cover the rest of the flow starting from "clicking" on the confirm email

## 2/A

The supplied listing is no more viable so I have choosen a different one, as perc the task description. The choosen listing’s link can be found in the (example) .env file. This is also the place to change the link give, the listing is not available again.

5 features to test on the listing details page:

* Problem with listing report functionality
* Carvertical / TotalCar redirection
* Car details against a constant database
* Financing calculator / redirection
* Seller’s further listings

## 2/B

Please check the source code for the implementation. – listing-details-and-parking-lot.spec.ts

In this test the user logs in (currently via UI, but it could be a lot faster with API login), loads up the listing provided in the .env file, presses the „Parkolóba rakom” button, and checks

* if the number besides „Parkoló” has increased or not,
* „Parkolóba rakom” button has disappeared
* „Kiveszem a parkolóból” button has appeared
* on the „Pakoló” page the correct listing (based on ad-id) has appeared

As a sort of teardown the listing is removed from the „Parkoló”. As such the test is executable multiple times, and will not face with test data issue.

The above assertions are intuitive, again, do not cover full functionality. For that, a meeting would be needed from other stakeholders (product, devs, BIs, etc) to have a common ground what should happen in this scenario.

## 2/C

For the implementation I choose the first two list item, as they are quite important from the business point of view.

Implementing these tests were not hard, but I once again faced an issue with the I am not a robot check.