**Anglo- SDET Assessment**

Please complete following exercises. You are free to complete both the exercises.

* We expect UI Test exercise to be in BDD framework, please use Cucumber or Specflow to write Feature files and test.
* We prefer UI test written in - Cucumber BDD. Javascript/Typescript using Playwright/Selenium/WebdriverIo.
* But you can use other language if you are comfortable with.
* We would like to see use of Implementation of Context injections to store data properties and access them in binding classes.
* Once assessment is completed, please upload project to GitHub with instructions on how to run tests. You can make github repository as public and share link with us.

**Exercise 1- UI Test**

* Task – Visit public website[**https://automationteststore.com/**](https://automationteststore.com/)
* Write 3-4 testcase on this using BDD style with basic framework structure.  
  which contains   
  1. Featurefile

2. StepDefination file

3. Hooks

4. Pageobjects

You can write 2-3 test scenarios for E.g.

- Filter products based on Type-Brand

- Add Products to cart- Include Sale Items

- View and verify Cart

- Make sure acceptance test should have Test data and expected results.

**Exercise 2- API Test**

**Assessment Notes**

* The attachment contains solution ShowroomService.
* ShowroomService solution contains the API with the endpoint which needs to be tested
* In order to run this ShowroomService solution just load it up in Visual studio and press F5
* ShowroomService contains following GET endpoint /api/cars/{type}
* Please create separate solution/project to write API test.

**Assessment Task:**

* We would like a BDD test for the endpoint in the API Provided and the example Acceptance criteria below, to a standard that you would consider complete for a production system

**Example Acceptance criteria**

* Endpoint to fetch a cars of specified type
* route should be /api/cars/{type}
* Valid car types are "Saloon", "SUV", "Hatchback"
* Returns status code 200 if cars of specified type exist along with list of cars of specified type
* Test which will compare and verify output of cars response based on Car type.
* Returns status code 404 if cars of specified type do not exist
* Any failure scenario’s you will consider to write.
* In Short consider this service to test fully for production release and what test you can automate to satisfy it’s release to production.

**Example response:**

[

{

"make": "BMW",

"model": "3.25i",

"year": "2013",

"type": "Saloon",

"zeroToSixty": 9.5,

"price": 15000

}

]