TOQIO – QA ENGINEER - CASE STUDY TEST

PHASE 1 – FUNCTIONAL TEST SELECTION AND CASE WRITING:

After taking the time to analyze your Toqio website (<https://toqio.co/>), in order to make sure that the upload to production of the new version has been correct, I consider the following two test cases most important.

* **User shall be able to request a demo from your website**
* **User shall be able to navigate through your website without getting any kind of error (like page not found, access denied and so on)**

Below, you can see how I wrote them using **gherkin language**:

**Feature: Toqio Website - Homepage**

**Verify request demo form and if there are any broken links**

**Background: Visit Toqio website**

**Given that I'm visiting toqio website**

**Scenario: User shall be able to request a demo from your website**

**When I see the request demo button and I click on it**

**Then I have been redirect to the contact page**

**And I can send an email to the company by submitting the form**

**Scenario: User shall be able to navigate through your website without getting any kinds of error (like page not found, access denied and so on)**

**Then I want to verify if all links are reachable**

I chose these two test cases because since the company offers a customisable platform for possible future customers and by filling out a form you can request a demo, for me this functionality is essential for your site.

Furthermore, if the site has broken links (such as page not found, access not allowed, server not reachable, etc.), the user opts for another service and no longer uses your website.

-----------------------------------------------------------------------------------------------------

PHASE 2 – AUTOMATION OF THE SELECTED TESTS:

For the automation of the selected tests cases I used **Cypress** framework with a couple of plugins, which are:

* **cypress-cucumber-preprocessor**
* **cypress-iframe**

And I run them on both **Chrome** and **Electron** browsers.

-----------------------------------------------------------------------------------------------------

Why I choose Cypress as a test framework?

* I feel more confident because I started using it in 2023
* It’s widely adopted by a lot of companies
* It’s easy to setup and configure
* There is a lot of documentation and good community

-----------------------------------------------------------------------------------------------------

How to launch them?

* Unzip the folder that I sent to you by email
* If you don’t have, download and install
  + **Nodejs** > <https://nodejs.org/en>
* Then check the installation
  + **node -v**
  + **npm -v**
* If you don’t have it, download and install **VSCode**
* Launch it and open the unzipped folder
* Click on the **package.json** file to see all the dependencies that need to be installed in order to run test cases
* Open the terminal from VSCode and use this command to install all dependencies
  + **npm install**
* Open cypress using this command
  + **npx cypress open**
* Then select the browser that you prefer (I have tried to run them on both browsers, Chrome and Electron)
* Then you see that there is one **.feature file** listed called “**Homepage.feature**”
* Click on it to run the test cases
* If you see both with a green flag, both test cases are passed successfully
* Otherwise you can see cypress error message

-----------------------------------------------------------------------------------------------------

How the project is structured?

* **feature** = this folder contains the **.feature** file, on which test cases / scenarios are written using **Gherkin** language, so everyone can read and understand the scope of them
* **pages** = this folder contains two **.cy.js** files, which are two **classes** with **static methods** that can be imported and called into the step definition file
* **pageSelectors** = this folder contains only **const** variables with all the css identifiers (like class, id, input type, input name)
* **step\_definitions** = this folder contains the **step file**, on which all scenario steps (so the keywords given, when, then, and) are mapped and pages are imported
* **fixtures** = actually the folder is empty, but it can be used to store **json** file that can be loaded inside test cases
* **support** = it contains two files **commands.js** and **e2e.js**, inside the command file I have only imported the **cypress-iframe plugin**
* **node\_modules** = this folder is the first folder that has been created when I initialized the project by executing the **npm init -y** command
* **cypress.config.js** = I use this file for setup and configure plugin inside the **setupNodeEvents** method and for configure some e2e parameters
* **package.json** = this file has been created when I initialized the project by executing the **npm init -y** command

-----------------------------------------------------------------------------------------------------

PHASE 3 – QA THEORY CASE:

From my experience, what the company can be adopts in order to reduce the number of bugs in production could be:

* **Working in Agile way**, so release a first version (MVP) of the product with some basic functionalities, got feedback from stakeholders and possible future end users and add a new piece of product at each iteration
* **Involve QA engineers in all meetings** (like sprint planning, sprint review, daily meeting, UI/UX call, etc…), so they know the product and what customers / stakeholders expect from it and they can also support R&D team and PM/PO
* **Setup a test automation project,** it can used to reduce time and to automate test cases on which actions are always the same and repeated every time
* **Put requirements in a shared place, use it as reference,** this is a good pratice because everybody can access to it, work items become more clean and clear with very few details and we don’t need to open each items for updating the requirement