Use a web testing framework of your choosing, e.g., webdriverIO, Playwright, Cypress, etc. Complete the following test cases using best practices such as page object model approach. The Web Test and Api Test do not necessarily need to be on the same framework. Add comments for areas you would improve or handle differently if time permits.

Test Case 1

1. Go got https://start.duckduckgo.com/
2. Search for the word: android
3. Assert that each entry in the results has the word `android` in the title

Code:

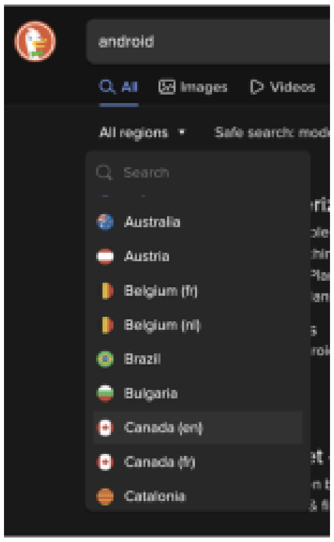
```
const assert = require('assert'); // Import the assert module
const SearchPage = require('../pageobjects/search.page')

describe('Make a Search using the DuckDuckGo page', async () => {

    beforeEach(async () => {
        await SearchPage.openPage(); // Open DuckDuckGo homepage
        await SearchPage.makeASearch('android') //make a search by the word
android
    });
```

Test 1:

```
it('should verify search results contain the term "android"', async () => {

        //verifying the results title
        await SearchPage.checkSearchTitles()

    });
```

Test Case 2

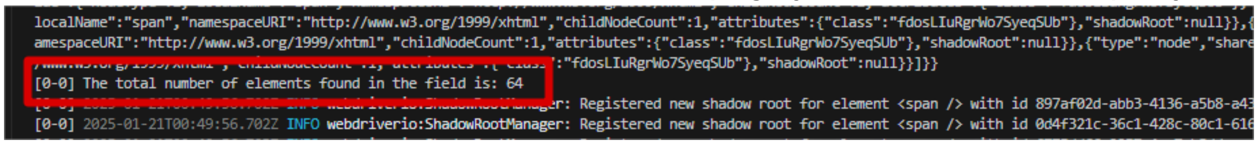
1. From the results page click All Regions
2. A modal will be displayed
3. Retrieve all elements and get a total count
4. Assert that the total count is greater than 10



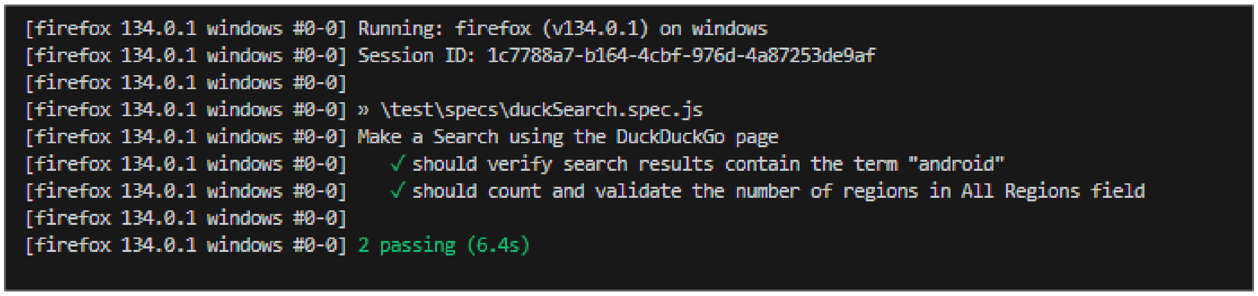
Test 2:

```
    it('should count and validate the number of regions in All Regions
field', async () => {

        // Retrieve and count the total number of elements in the "All
Regions" field
       await SearchPage.getAllRegionCount();



        });
```

The full code can be found on the files search.page.js and duchSearch.spec.js



```
localName":"span","namespaceURI":"http://www.w3.org/1999/xhtml","childNodeCount":1,"attributes":{"class":"fdosLIuRgrWo7SyeqSUb"},"shadowRoot":null}},{
amespaceURI":"http://www.w3.org/1999/xhtml","childNodeCount":1,"attributes":{"class":"fdosLIuRgrWo7SyeqSUb"},"shadowRoot":null}},{"type":"node","share
                                                     ":"fdosLIuRgrWo7SyeqSUb"},"shadowRoot":null}}]}}
[0-0] The total number of elements found in the field is: 64
                                                     r: Registered new shadow root for element <span /> with id 897af02d-abb3-4136-a5b8-a43
[0-0] 2025-01-21T00:49:56.702Z INFO webdriverio:ShadowRootManager: Registered new shadow root for element <span /> with id 0d4f321c-36c1-428c-80c1-616
```



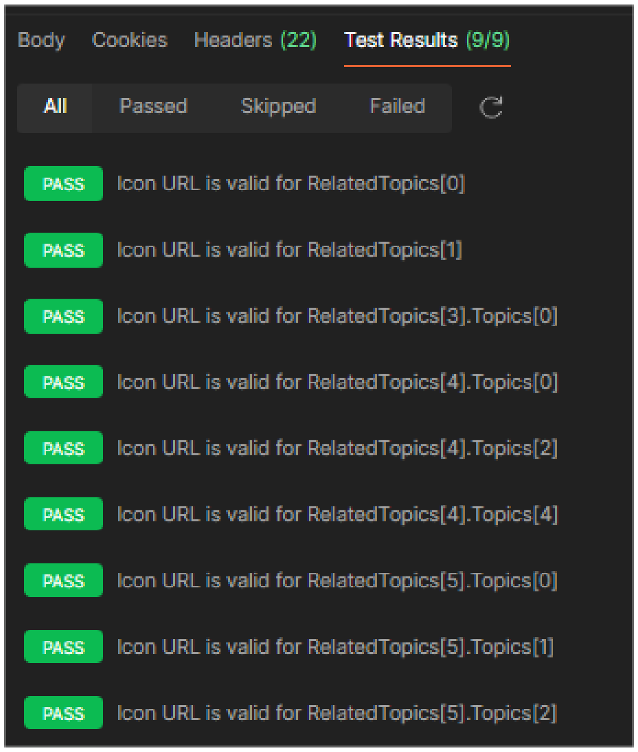
```
[firefox 134.0.1 windows #0-0] Running: firefox (v134.0.1) on windows
[firefox 134.0.1 windows #0-0] Session ID: 1c7788a7-b164-4cbf-976d-4a87253de9af
[firefox 134.0.1 windows #0-0]
[firefox 134.0.1 windows #0-0] » \test\specs\duckSearch.spec.js
[firefox 134.0.1 windows #0-0] Make a Search using the DuckDuckGo page
[firefox 134.0.1 windows #0-0]    ✓ should verify search results contain the term "android"
[firefox 134.0.1 windows #0-0]    ✓ should count and validate the number of regions in All Regions field
[firefox 134.0.1 windows #0-0]
[firefox 134.0.1 windows #0-0] 2 passing (6.4s)
```

Test Case 3 - Handling a JSON response

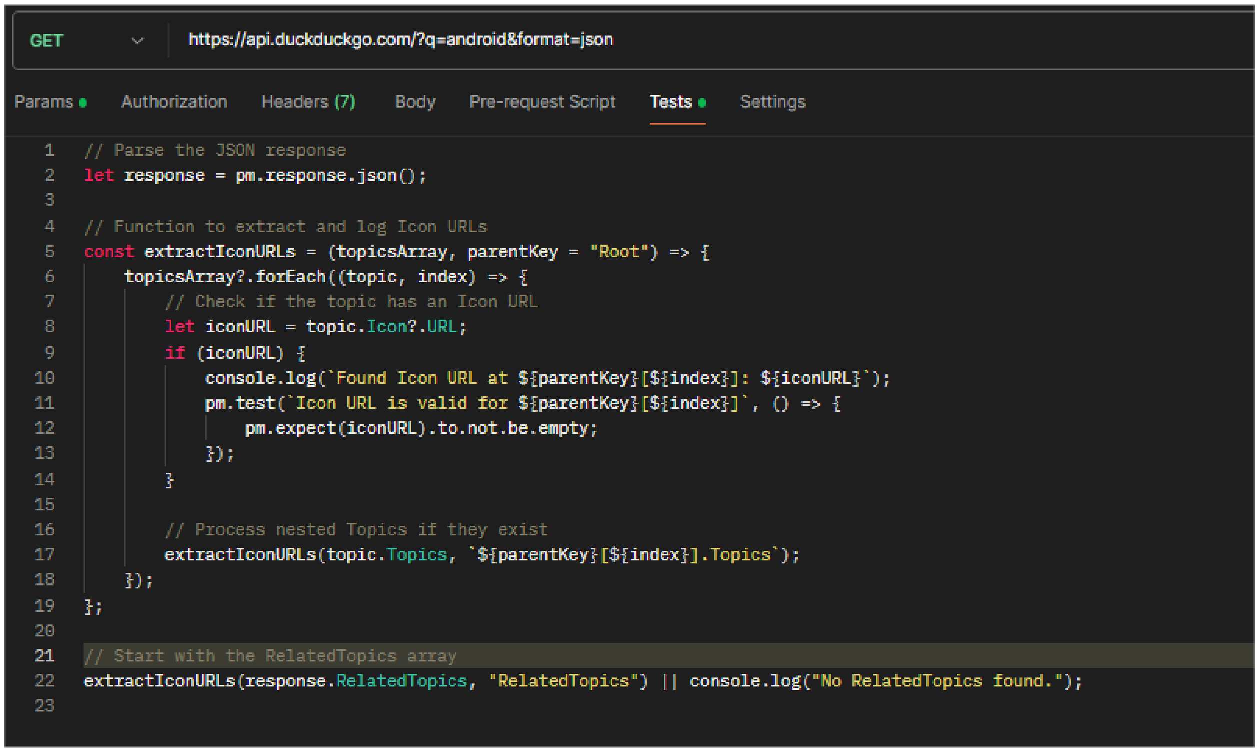
There's no UI for this one. You can use the same framework you used for the previous test cases or use a different framework/library. Implement it with a solution you are most comfortable with.

* Make a call to the following URL: https://api.duckduckgo.com/?q=android&format=json *
Handle the response
* Print out the value of the Icon URL if it's not null. The data you are working with will look something like:

```
"Icon": {
 "Height": "",
 "URL": "/i/d58b5fe4.png",
 "Width": ""
},
```



```
PASS Icon URL is valid for RelatedTopics[0]
PASS Icon URL is valid for RelatedTopics[1]
PASS Icon URL is valid for RelatedTopics[3].Topics[0]
PASS Icon URL is valid for RelatedTopics[4].Topics[0]
PASS Icon URL is valid for RelatedTopics[4].Topics[2]
PASS Icon URL is valid for RelatedTopics[4].Topics[4]
PASS Icon URL is valid for RelatedTopics[5].Topics[0]
PASS Icon URL is valid for RelatedTopics[5].Topics[1]
PASS Icon URL is valid for RelatedTopics[5].Topics[2]
```



```javascript
// Parse the JSON response
let response = pm.response.json();

// Function to extract and log Icon URLs
const extractIconURLs = (topicsArray, parentKey = "Root") => {
    topicsArray?.forEach((topic, index) => {
        // Check if the topic has an Icon URL
        let iconURL = topic.Icon?.URL;
        if (iconURL) {
            console.log(`Found Icon URL at ${parentKey}[${index}]: ${iconURL}`);
            pm.test(`Icon URL is valid for ${parentKey}[${index}]`, () => {
                pm.expect(iconURL).to.not.be.empty;
            });
        }

        // Process nested Topics if they exist
        extractIconURLs(topic.Topics, `${parentKey}[${index}].Topics`);
    });
};

// Start with the RelatedTopics array
extractIconURLs(response.RelatedTopics, "RelatedTopics") || console.log("No RelatedTopics found.");

```