The entire project was completed in 14.5 hours, distributed as follows:

* Review of requirements and exploratory testing (1.5 hours): I familiarized myself with the website, its features, layout, and user flow, focusing on the Product Search and Store Search features.
* Test environment setup (2 hours): I installed and configured the Cypress Testing Framework, Cypress Dashboard, and the corresponding GitHub repositories.
* Scenario selection and replay (1 hour)
* Coding of framework and page objects (5 hours): This included creating

**mainPage.po.js** for interacting with the Maax.com main page,

**searchResult.po.js** for interacting with the product search result page (to verify product details, filter search results, etc.),

**whereToBuy.po.js** for interactions with the 'Where to Buy' page result page (searching by address, using the 'Find My Location' feature, and verifying store details).

* Documentation (1.5 hour)
* Coding of test scenarios and test data (3.5 hours)

The **search\_products\_scenario.cy.js** file contains three scenarios:

1. Search for a single product and verify product details:

Read test data from fixture file (**productDetails.json**), which contains details of four products.

Foreach of these four products:

Visit Maax.com

Perform a search using the product ID and verify that only one product is listed in the search results.

Verify the product details (name, dimensions, price, colors, characteristics).

1. Search using a keyword and verify product category filters and product count:

Read test data from fixture file (**searchDetails.json**), which contains a single search scenario that should return 17 results. The expected number of products is also stored for the following filters: **productCategories.ShowerDoor** and **productCategories.MedicineCabinet**.

Visit Maax.com

Perform a search using a keyword and verify the total product count.

Select the filter **productCategories.ShowerDoor** and verify the product count. If the count is greater than 12, verify that the 'Load More' button is displayed.

Clear the filter and verify the product count.

Select the filter **productCategories.MedicineCabinet** and verify the product count. If the count is greater than 12, verify that the 'Load More' button is displayed.

1. Search using a keyword and verify product series filters and product count:

Read test data from fixture file (**searchDetails.json**), which contains a single search scenario that should return 17 results. The expected number of products is also stored for the following filters: **productSeries.Collection** and **productSeries.MAAX**.

Visit Maax.com

Perform a search using a keyword and verify the total product count.

Select the filter **productSeries.Collection** and verify the product count. If the count is greater than 12, verify that the 'Load More' button is displayed.

Clear the filter and verify the product count.

Select the filter **productSeries.Maax** and verify the product count. If the count is greater than 12, verify that the 'Load More' button is displayed.

The **where\_to\_buy\_scenario.cy.js** file contains two scenarios:

1. Searching for new addresses sequentially - Actual Bug (minor):

Read test addresses from the **locationSearchDetails** fixture file. Each address has a total number of stores associated with it.

Visit Maax.com's 'Where to Buy' page.

For each address:

Input the address and select it from the dropdown.

Verify the store count. If the count is more than 6, verify that the 'Next Page' button is displayed.

1. Find a store using geolocation and verify the details on the first page:

Read two sample geolocations and the store count for each location from **locationSearchDetails.json**.

For each geolocation:

Visit Maax.com's 'Where to Buy' page and update the browser’s geolocation coordinates with the test coordinates.

Click on 'Get My Location' and wait for the store list to load.

Verify the store count. If the count is more than 6, verify that the 'Next Page' button is displayed.

Verify the store details on the first page.

Reasons for selecting these scenarios:

* I aimed to demonstrate a broad range of skills within a limited time.
* As indicated in the assessment document, I focused on the Product Search and Store Location Search areas. I guessed 4 common use of these features,
* I intentionally included both successful and failing test scripts to illustrate the discovery of bugs.
* I wanted to gain experience with using geolocation with Cypress and testing the 'Get My Location' feature on Maax.com.

Note:

* The code was written in VS Code and formatted using Prettier.

Challenges & Lessons learned:

1. Familiarizing myself with an unfamiliar website that had numerous errors, though not always reproducible.
2. Finding a solution for geolocation (new information).
3. Understanding the search logic and filter behaviors took time.
4. Generating test data from the UI (product details, store details, etc.) was time-consuming.
5. An unnecessary fix caused a rollback to Cypress 4.x after installing a plugin to control browser permissions,
6. A Dell docking station is not ideal for a Lenovo laptop.
7. Automation is enjoyable and a personal passion.

Other bugs found:

1. Using double quotation marks in the search string results in a blank page and console errors.
2. When searching for 'Medicine Cabinet door' with a total of 17 products found:
   * Selecting the filter **productSeries.collection** results in 15 products listed.
   * Clicking on 'View More' still only shows 15 products.
   * Changing the sort order to 'Price Ascending' results in only 3 products being displayed instead of 15.
3. There were other filter bugs, but they could no longer be reproduced.