Paola Pierola Deel 10.12.2023

12 OCTOBER

Exercise 1:

Unix command ’we’ takes a text file as input and tells you how many characters, words and lines it has. A team is working on an improved version that adds a new function: it counts words and their frequency. I.e.: How many times each word appears in the text.

Example: given the input 'This is a line\nAnd this is another,
The new ‘we’ provides the following output:

lines = 2
words = 8
frequency = 'this' (2), 'is' (2), ’a' (1), 'line'(1), 'and' (1),
another (1)

As the list of frequencies can be too long, it is determined that the first 5 more frequent words will be displayed.
In case there are less than 5 words, the existing ones will be displayed.

According to the changes mentioned above, the frequency value is renamed as 'top-5', thus changing the output of the prior example as follows:

lines = 2
words = 8

top-5 = 'is' (2), 'this' (2), 'a' (1), 'and' (1), 'line'(1)

a. Detail of test cases (inputs) that would be generated by testing top-5 functionality
b. Mention any assumption you made or any information that you feel is missing.

Solution – Exercise 1: A and B:

Input:

This is a line
And this is another

Assumptions: **Top-5: String will start at the 5th position.**

Top-5 String:

is a line
And this is another

Requirements: **Unix**

Test Case 1:
1. Write the following command: wc -l
2. Verify that the script is executed.
3. Verify that the output is: 2

PASS/FAIL Criteria:

Test case will PASS if the output is 2. Since the number of lines is 2.
Test case will FAIL otherwise.

Test Case 2:

1. Write the following command: wc -w
2. Verify that the script is executed.
3. Verify that the output is: 7

PASS/FAIL Criteria:

Test case will PASS if the output is 7. Since the number o words is 7.
Test case will FAIL otherwise.

Test Case 3:

1. Write the following command: wc -C
2. Verify that the script is executed.
3. Verify that the output is the following:

is(2)
a(1)
line(1)
and(1)
this(1)
another(1)

PASS/FAIL Criteria:

Test case will PASS if the output is:
is(2)
a(1)
line(1)
and(1)
this(1)
another(1). This will give you the frequency of how many times the word is displayed in the string.
Test case will FAIL otherwise.

Exercise 2:

The following code implements part of a purchase order system. It is made up of a component that queues purchase orders as they come in, and another component that takes those PO andprocesses them.
Each of these parts runs in a thread or separate procedure.
The queue has a max limit and thus, in the event it is full, no other order should be added untilspace is freed. Likewise, no order should be processed if the queue is empty.
To achieve this, the sleep()/wakeup() primitives are used; they stop/restart threadsexecution.
sleep() blocks a thread until it receives a wakeup():

```
thread place_order(data)
{
    while (true)
    {
        order = OrderFactory.produceOrder(data);
        if (self.queue.length() == MAX_QUEUE_SIZE)
        {
            // queue is full. Do nothing until somebody wakes us up
            sleep();
        }
        self.queue.insert(order);
        if (self.queue.size > 0)
        {
            // we have at least 1 element to process, notify the other component
            wakeup(process_order);
        }
    }
}
```

```
thread process_order()
{
    while (true)
    {
        if (self.queue.length() == 0)
        {
            // queue is empty, block until there is at least 1 element
            sleep();
        }
        // if the queue was full, and we just made room, notify the producer
        if (self.queue.length() == MAX_QUEUE_SIZE - 1)
        {
            wakeup(place_order);
        }
        order = self.queue.pop();
        do_actual_processing(order);
    }
}
```

a. Assume that there are automated tests that take a string describing the operation sequenceof the two components: "place_order(data1), place_order(data2), process_order(), process order(),.. " (or in its abbreviated version: "P, P, C, C, ...")
Describe what would be the most relevant situations (test cases) to be tested. For these testcases, mention the input that generates them (assuming MAX QUEUE SIZE=4)

b. Analyze the previous pseudocode, and identify errors (if any). In case of no error detected,but you would like to suggest improvements, mention them.

c. If there are N and N instead of 1 order generator and 1 order processing, sharing the same queue, what new situations that prior tests do not cover should be considered? Detail how you would implement tests and describe a typical error.

Solution – Exercise 1: A and B:

Solution A:

Test Case 1

Input: Have two PlaceOrder at the same time.

1. Execute place_order(data4) at the time: 00:00:00 AM
2. Execute place_order(data5) at the time: 00:00:00 AM
3. Execute place_order(data4)
4. Execute place_order(data5)
5. Verify that the application sets a token for data4 and data 5

PASS/FAIL Criteria:
Pass:
Order data4 has to be be a TokenA, and order data5 has to have a TokenB.
Fail:
When order data4 and order data5 have the same Token.
When TokenA is the same as TokenB.

Test Case 2

Input: MAX_QUEUE_SIZE = 4

1. Execute place_order(data1)
2. Execute place_order(data2)
3. Execute place_order(data3)
4. Execute place_order(data4)
5. Execute process_order(data1)
6. Execute process_order(data2)
7. Execute process_order(data3)
8. Execute process_order(data4)
9. Verify that the following orders have been executed: data1, data2, data3, and data4.

PASS/FAIL Criteria:
PASS
When the maximum number in queue is 4.
FAIL

When the maximum number in queue exceeds 4.

Solution B:

Add a procedure to give a token for concurrency cases.

Add a procedure to set priority for very large orders.

Add a procedure to set a flag when user abandons order.

Add a procedure to set a flag when user is idle.

Add a procedure when there is a bottleneck. This procedure should be setting priorities and Tokens for each order. So, that the server does not collapse.

Solution C :

I think that Concurrency must be considered:

Deleted updates. It is possible for two orders, A and B, to be placed at the same time. A's change will be overwritten if B updates the order after A.

Data that has not been committed. An order could be updated by order A, and order B might be placed before it is committed. The order B may therefore be based on incorrect information if A cancels that update.

singular readings. A order may be processed by Application before it handles other requests. Order B can be processed while waiting for order A to be completed, then commits the modification.

Exercise 3:

Once you completed the preparation steps mentioned at the beginning of this document, youare required to write the following automated tests:

URL: https://qrowth.deel.training/dev/salary-insights

Normal (Happy-path) flows with the following data:

- Accountant, Brazil
- QA Engineer, Canada
- Software Engineer, Japan

a. Automated tests should run and pass.
b. Prepare yourself to discuss additional improvements that can be made to your solution.

Solution:

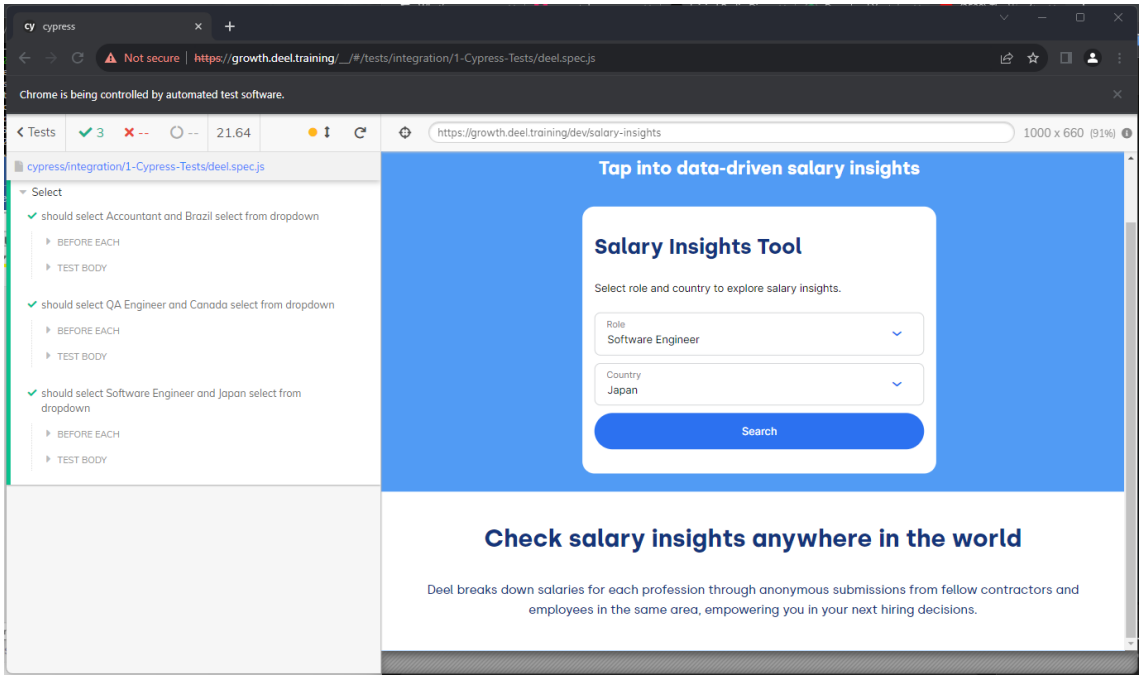
Please refer to the code: https://github.com/poletpierola/deel

a. Also, following code contains all Tests and all Tests Pass:

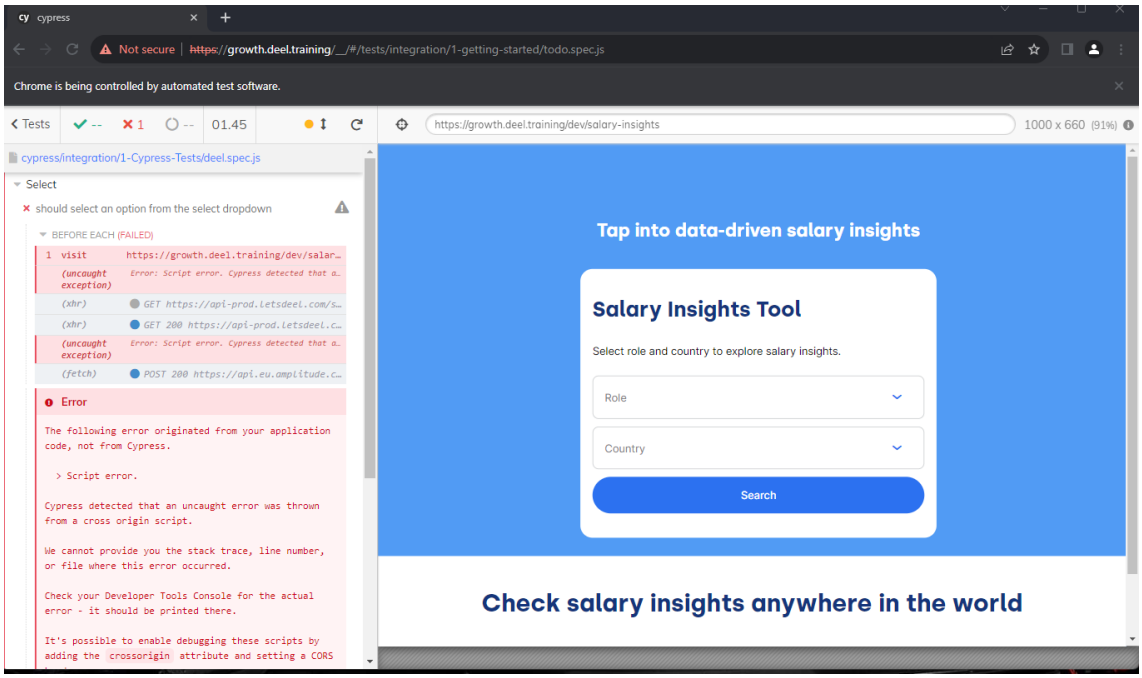
cypress > integration > 1-Cypress-Tests > deel.spec.js > ...

```
// deel.spec.js created with Cypress
//
// Start writing your Cypress tests below!
// If you're unfamiliar with how Cypress works,
// check out the link below and learn how to write your first test:
// https://on.cypress.io/writing-first-test
describe("Select", () => {

    beforeEach(() => {
      //To solve the uncaught exception error
      Cypress.on('uncaught:exception', (err, runnable) => {
        return false
        })
        cy.visit('https://growth.deel.training/dev/salary-insights') // yup all good
        cy.wait(5000);


    })


    //Selects Accountant from Role dropdown and Brazil from Coutry dropdown
    it('should select Accountant and Brazil select from dropdown', () => {

      //Role
      cy.get('#mui-2').click()

      //Accountant
      cy.get('#mui-2-option-0').click()

      //Brazil
      cy.get('#mui-4').click()
      cy.get('#mui-4-option-3').click()


    });

    //Selects Accountant from Role dropdown and Brazil from Coutry dropdown
    it('should select QA Engineer and Canada select from dropdown', () => {
      //Role
      cy.get('#mui-2').click()
      //QA Engineer
      cy.get('#mui-2-option-104').click()

      //Canada
      cy.get('#mui-4').click()
      cy.get('#mui-4-option-4').click()

    });

    //Selects Accountant from Role dropdown and Brazil from Coutry dropdown
    it('should select Software Engineer and Japan select from dropdown', () => {
      //Role
      cy.get('#mui-2').click()

      //QA Engineer
      cy.get('#mui-2-option-124').click()

      //Brazil
      cy.get('#mui-4').click()
      cy.get('#mui-4-option-13').click()

    });


});
```

Execution:



Issues and Exceptions that were troubleshooted:



How Exception was fixed:

```
//To solve the uncaught exception error
Cypress.on('uncaught:exception', (err, runnable) => {
  return false
  })
  cy.visit('https://growth.deel.training/dev/salary-insights') // yup all good
  cy.wait(5000);
```

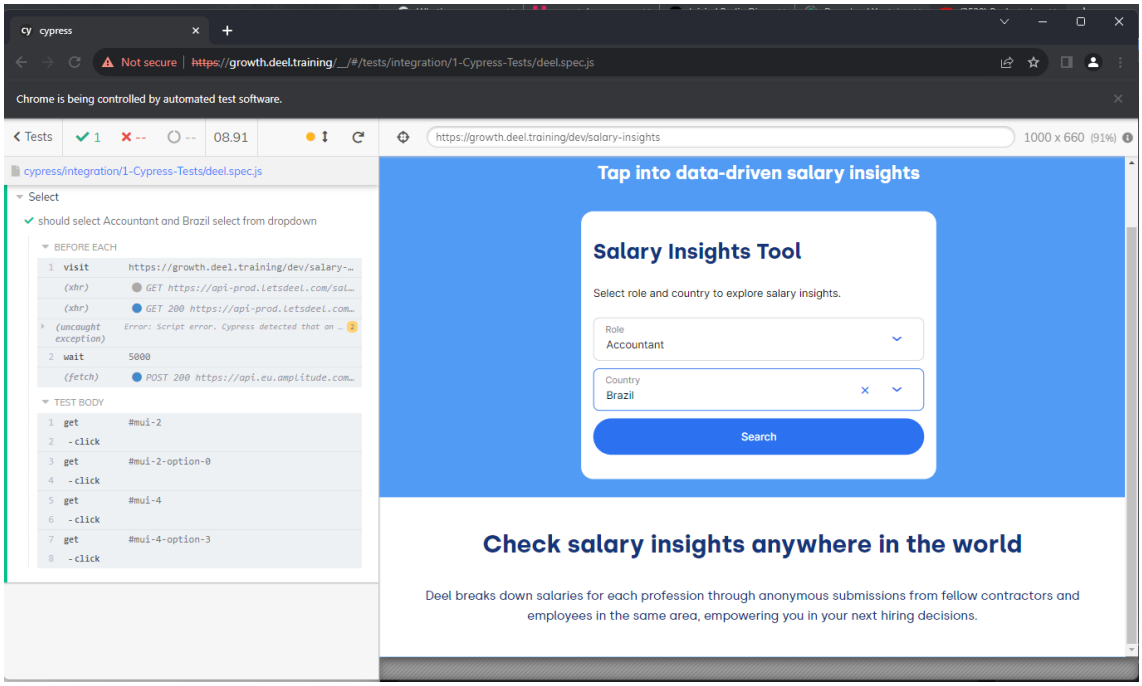
Test 1:

Role: Accountant
Country: Brazil

Code Test 1:

```
//Selects Accountant from Role dropdown and Brazil from Coutry dropdown
it('should select Accountant and Brazil select from dropdown', () => {

  //Role
  cy.get('#mui-2').click()

  //Accountant
  cy.get('#mui-2-option-0').click()

  //Brazil
  cy.get('#mui-4').click()
  cy.get('#mui-4-option-3').click()


});
```

Execution Test 1:



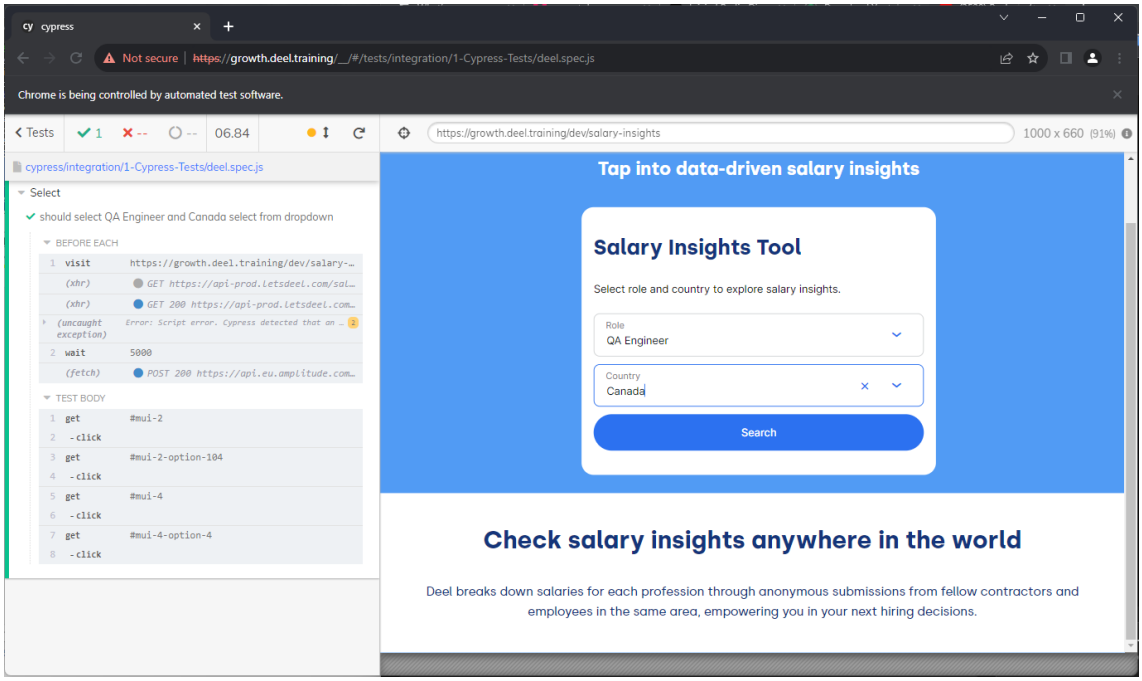
Test 2:

Role: QA Engineer
Country: Canada

Code Test 2:

```
//Selects Accountant from Role dropdown and Brazil from Coutry dropdown
it('should select QA Engineer and Canada select from dropdown', () => {
  //Role
  cy.get('#mui-2').click()
  //QA Engineer
  cy.get('#mui-2-option-104').click()

  //Canada
  cy.get('#mui-4').click()
  cy.get('#mui-4-option-4').click()

});
```

Execution Test 2:



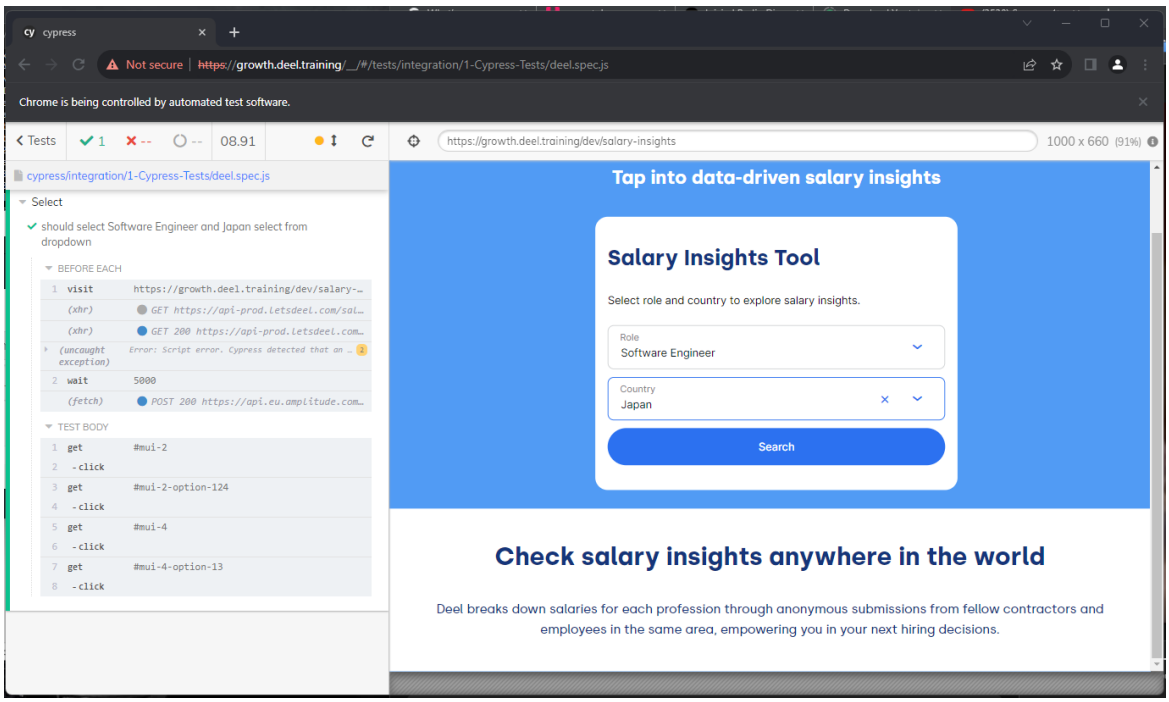
Test 3:

Role: Software Engineer
Country: Japan

Code Test 3:

```
//Selects Accountant from Role dropdown and Brazil from Coutry dropdown
it('should select Software Engineer and Japan select from dropdown', () => {
    //Role
    cy.get('#mui-2').click()

    //QA Engineer
    cy.get('#mui-2-option-124').click()

    //Brazil
    cy.get('#mui-4').click()
    cy.get('#mui-4-option-13').click()

});
```

Execution Test 3:



Please feel free to download the following files:

Tests Solution: deel.spec.js:

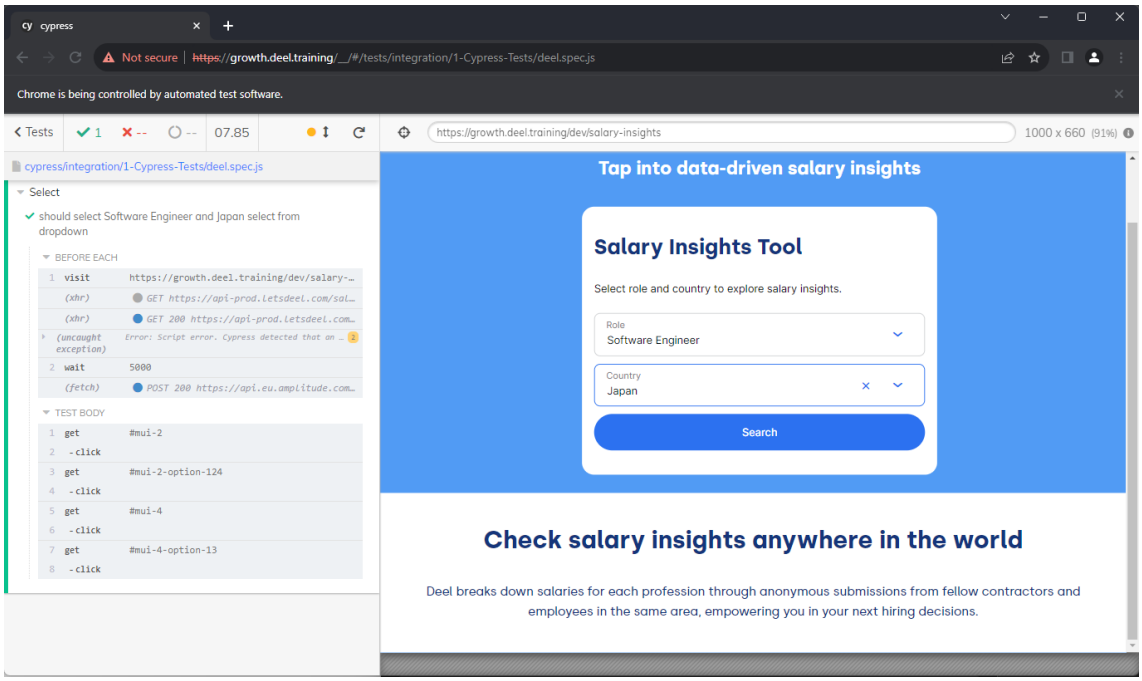
https://github.com/poletpierola/deel/blob/main/deel.spec.js

Tests being executed Video: 2023-10-12_13-34-43.mp4.

https://github.com/poletpierola/deel/blob/main/2023-10-12_13-34-43.mp4

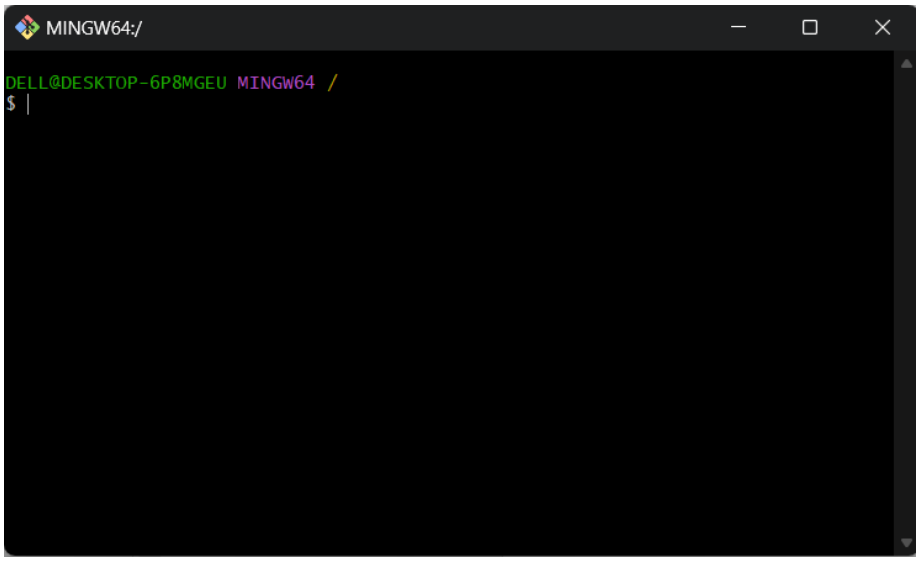
Paola Pierola Test PDF File: Paola Pierola Deel 10122023 .pdf. '

https://github.com/poletpierola/deel/blob/main/Paola%20Pierola%20Deel%20101220 23%20.pdf



GIT-BASH

1. Download Git-Bash from (Win-64Bits): https://github.com/git-for-windows/git/releases/download/v2.42.0.windows.2/Git-2.42.0.2-64-bit.exe
2. Install Git-Bash.
3. Execute Git-Bash.

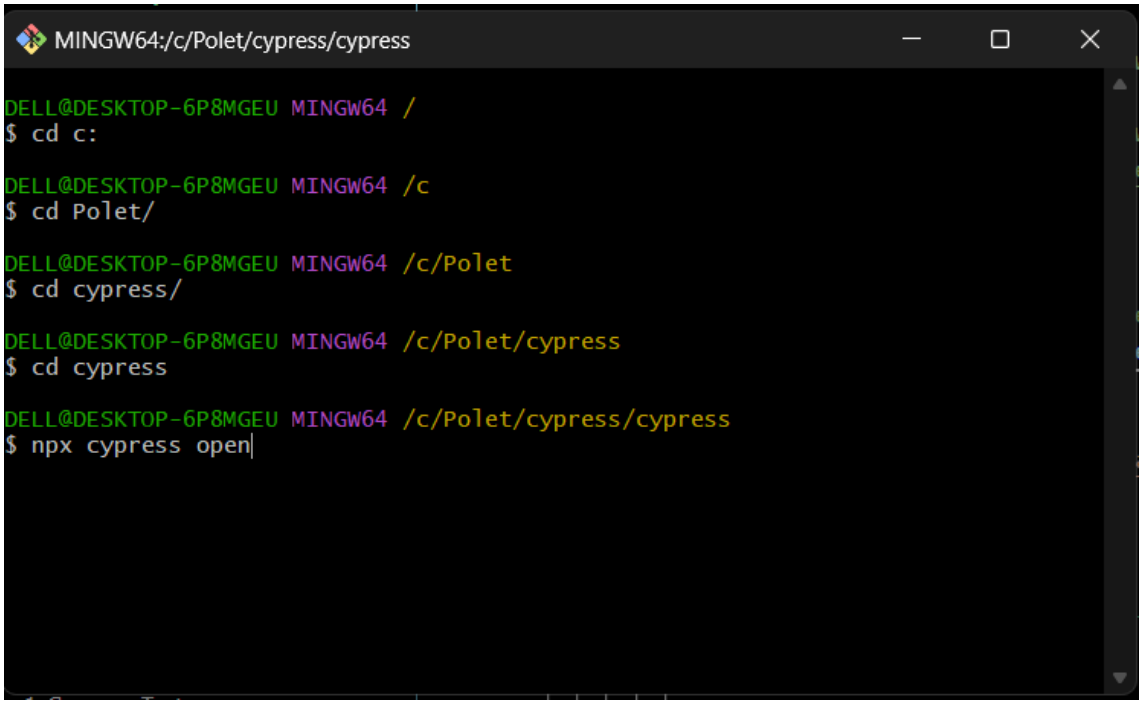


Cypress.io

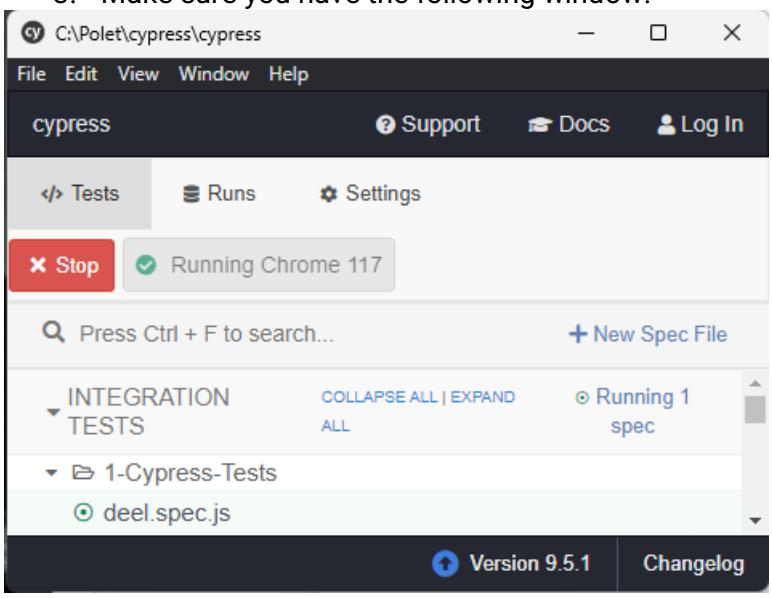
1. Go to the root where you need to install Cypress.io
2. Type: `npm install cypress --save-dev`

To execute Cypress Tests

1. Go where cypress has previously been installed.
2. Type `npx cypress open`



3. Make sure you have the following window:



4. Click on deel.spec.js
5. Then, all tests will begin executing automatically.