PA1417: Basic System Verification
Tutorial 3: GUI Testing with Cypress

Julian **Frattini**

BLEKINGE INSTITUTE OF TECHNOLOGY

Challenges of Test Automation

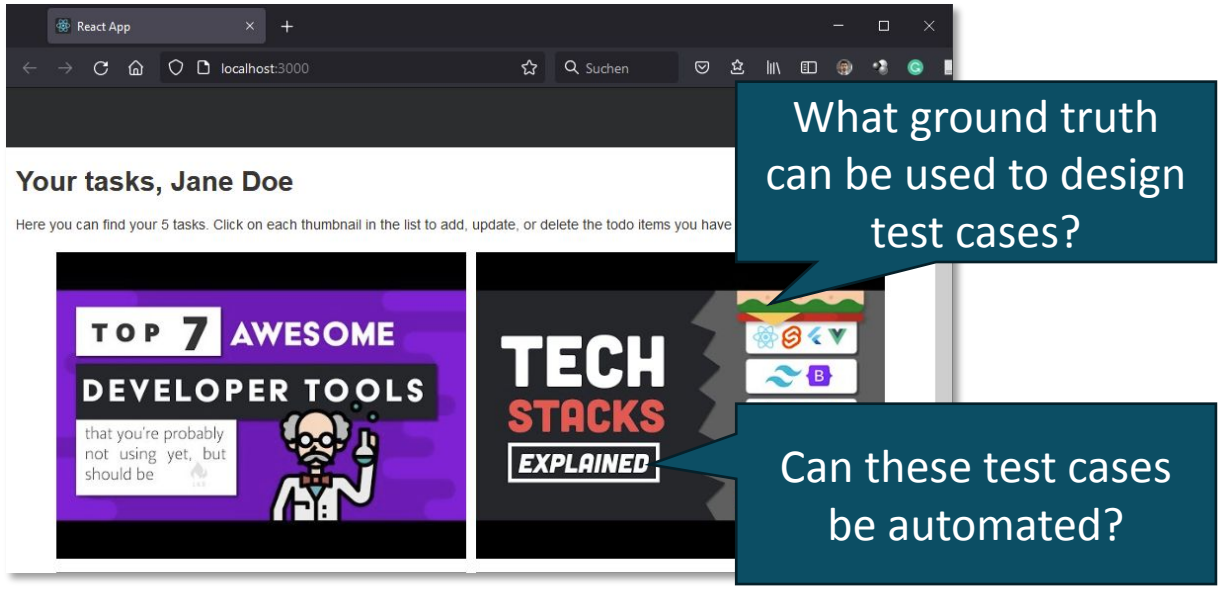
BLEKINGE INSTITUTE OF TECHNOLOGY

When testing (parts of the) **backend**, the interface of each function is clearly defined.

```
def get_user_by_email(self, email: str):
    """Given a valid email address of an existing account, return the user object contained in the
    database associated
    to that user. For now, do not assume that the email attribute is unique. Additionally print a
    warning message containing the email
    address if the search returns multiple users.

    parameters:
        email -- an email address string

    returns:
        user -
        that
        None -
    raises:
        ValueE
        <host:
        Except
    """
    if not re
        raise
    try:
        users
        if len
            re
        else:
            pr
            re
    except Exc
        raise
```

```
> python .\main.py
Connecting to collection user on MongoDB at url mongodb://localhost:27017
Connecting to collection task on MongoDB at url mongodb://localhost:27017
Connecting to collection video on MongoDB at url mongodb://localhost:27017
Connecting to collection todo on MongoDB at url mongodb://localhost:27017
Map([<Rule '/users/create' (POST, OPTIONS) -> user_blueprint.create_user>,
 <Rule '/tasks/create' (POST, OPTIONS) -> task_blueprint.create>,
 <Rule '/todos/create' (POST, OPTIONS) -> todo_blueprint.create>,
 <Rule '/users/all' (GET, HEAD, OPTIONS) -> user_blueprint.get_users>,
 <Rule '/populate' (POST, OPTIONS) -> populate>,
 <Rule '/' (GET, HEAD, OPTIONS) -> ping>,
 <Rule '/users/bymail/<email>' (GET, HEAD, OPTIONS) -> user_blueprint.get_user_by_mail>,
 <Rule '/tasks/ofuser/<id>' (GET, HEAD, OPTIONS) -> task_blueprint.get_tasks_of_user>,
 <Rule '/tasks/byid/<id>' (GET, HEAD, PUT, OPTIONS) -> task_blueprint.get>,
 <Rule '/todos/byid/<id>' (DELETE, GET, HEAD, PUT, OPTIONS) -> todo_blueprint.get_todo>,
 <Rule '/static/<filename>' (GET, HEAD, OPTIONS) -> static>,
 <Rule '/users/<id>' (GET, HEAD, PUT, OPTIONS) -> user_blueprint.get_user>])
 * Serving Flask app 'todoapp' (lazy loading)
 * Environment: production
```

When testing (parts of the) **frontend**, the interface of each function requires the identification of elements in a user interface.



Ground Truth for Test Design

BLEKINGE INSTITUTE OF TECHNOLOGY

To test the backend, we used docstrings as our oracle. For UI tests, we need to resort to a different ground truth documentation.

Backend Tests

```python
def hasAttribute(obj: dict, attribute: str):
    """Check whether a given dict contains a specific attribute

    attributes:
        obj -- a dict object
        attribute -- the key which potentially occurs in the obj dict

    returns:
        True -- if the dict contains the attribute as a key
        False -- if the dict does not contain the attribute as a key or is None
    """
    return (attribute in obj)
```

Frontend Tests

ID	R3UC1
Primary Actor	End user
Preconditions	-
Main Success Scenario	1. When the user enters the website, the system prompts the user to enter his/her credentials. 2. When the user enters a valid email-password combination and clicks “Login”, the system authenticates the user.
End Condition	The user is forwarded to an overview of the tasks associated to him/her.
Alternative Scenarios	2.b When the email is not yet registered, the system prompts the user to signup instead. 2.c When the password is incorrect, the system prompts the user to try again.

4-step Test Design Technique
Application to a Use Case

BLEKINGE INSTITUTE OF TECHNOLOGY

Quiz: Identify actions, expected outcomes, and conditions from use cases.

ID	R3UC1
Primary Actor	End user
Preconditions	-
Main Success Scenario	1. When the user enters the website, the system prompts the user to enter his/her credentials. 2. When the user enters a valid email-password combination and clicks "Login", the system authenticates the user.
End Condition	The user is forwarded to an overview of the tasks associated to him/her.
Alternative Scenarios	2.b When the email is not yet registered, the system prompts the user to signup instead. 2.c When the password is incorrect, the system prompts the user to try again.

The quiz will be available after the lecture at https://docs.google.com/forms/d/e/1FAIpQLSdTy_7kWzuyFLENkQ3dHDFO9BxwEt2lObB2AN_Dw3wondRGTg/viewform?usp=sf_link

4-step Test Design Technique
Application to a Use Case

BLEKINGE INSTITUTE OF TECHNOLOGY

Quiz: Identify actions, expected outcomes, and conditions from use cases.

ID	R3UC1
Primary Actor	End user
Preconditions	-
Main Success Scenario	1. When the user enters the website, the system prompts the user to enter his/her credentials. 2. When the user enters a valid email-password combination and clicks "Login", the system authenticates the user.
End Condition	The user is forwarded to an overview of the tasks associated to him/her.
Alternative Scenarios	2.b When the email is not yet registered, the system prompts the user to signup instead. 2.c When the password is incorrect, the system prompts the user to try again.

ID	-	User entering the website
1.1	-	The system prompts the user to enter his/her credentials

ID	email	Password	User clicks "Login"
2.1	registered	correct	system authenticates the user and forwards user to task overview
2.2	registered	incorrect	system prompts to try again
2.3	not registered	-	system prompts to signup

Testing Frameworks
Automating GUI Tests

BLEKINGE INSTITUTE OF TECHNOLOGY

There are several frameworks that allow the implementation of automatic GUI tests, for example:

1. Selenium (https://www.selenium.dev/)
2. Cypress (https://www.cypress.io/)

Test Level

BLEKINGE INSTITUTE OF TECHNOLOGY

Cypress supports both testing paradigms:

- **Component Tests**: unit tests of UI elements
- **E2E (end-to-end) Tests**: functional/scenario tests of the whole system

In assignment 4, we will focus on E2E tests.

Setting up Cypress

BLEKINGE INSTITUTE OF TECHNOLOGY

Requirements: the npx package must be installed (npm install npx), which usually comes automatically with npm

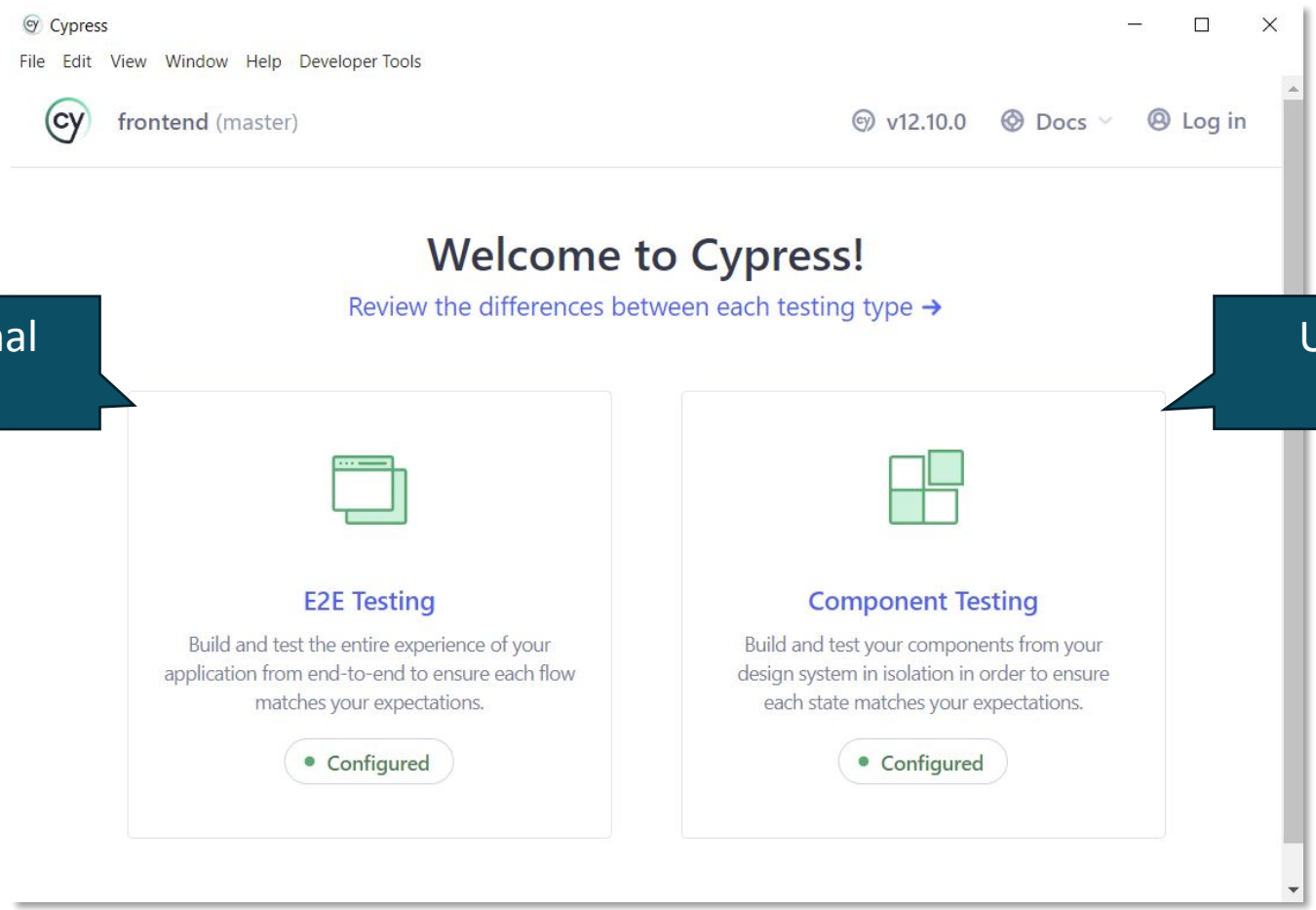
Setup:

1. Ensure that the system is running (either locally or via Docker)
2. Within the *frontend* folder, install cypress via `npm install -D cypress`
3. Open the cypress test runner via `npx cypress open`

Step 2 has already been prepared in the current system: when you set up the system via `npm install`, cypress was included.

Cypress Opening Window

BLEKINGE INSTITUTE OF TECHNOLOGY



Scenario/functional tests

Test Scope

GUI

Server

Database

Unit Tests on GUI Elements

Test Scope

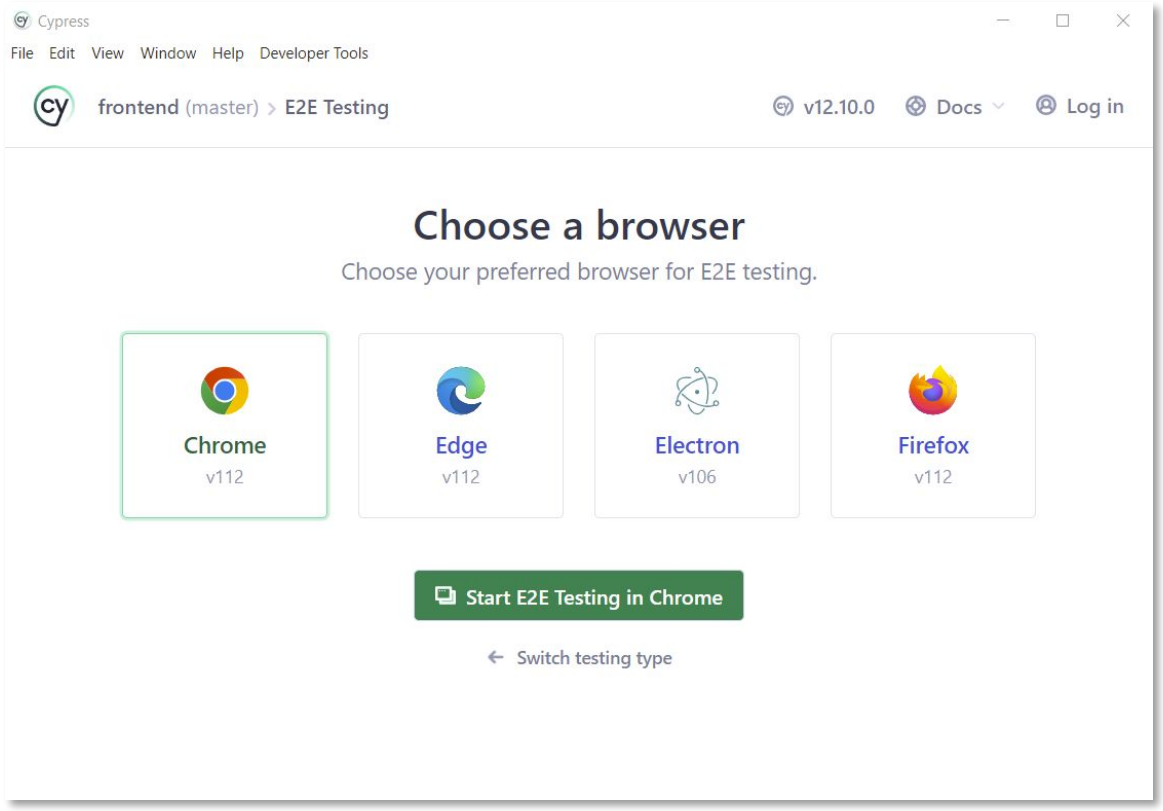
GUI

Server

Database

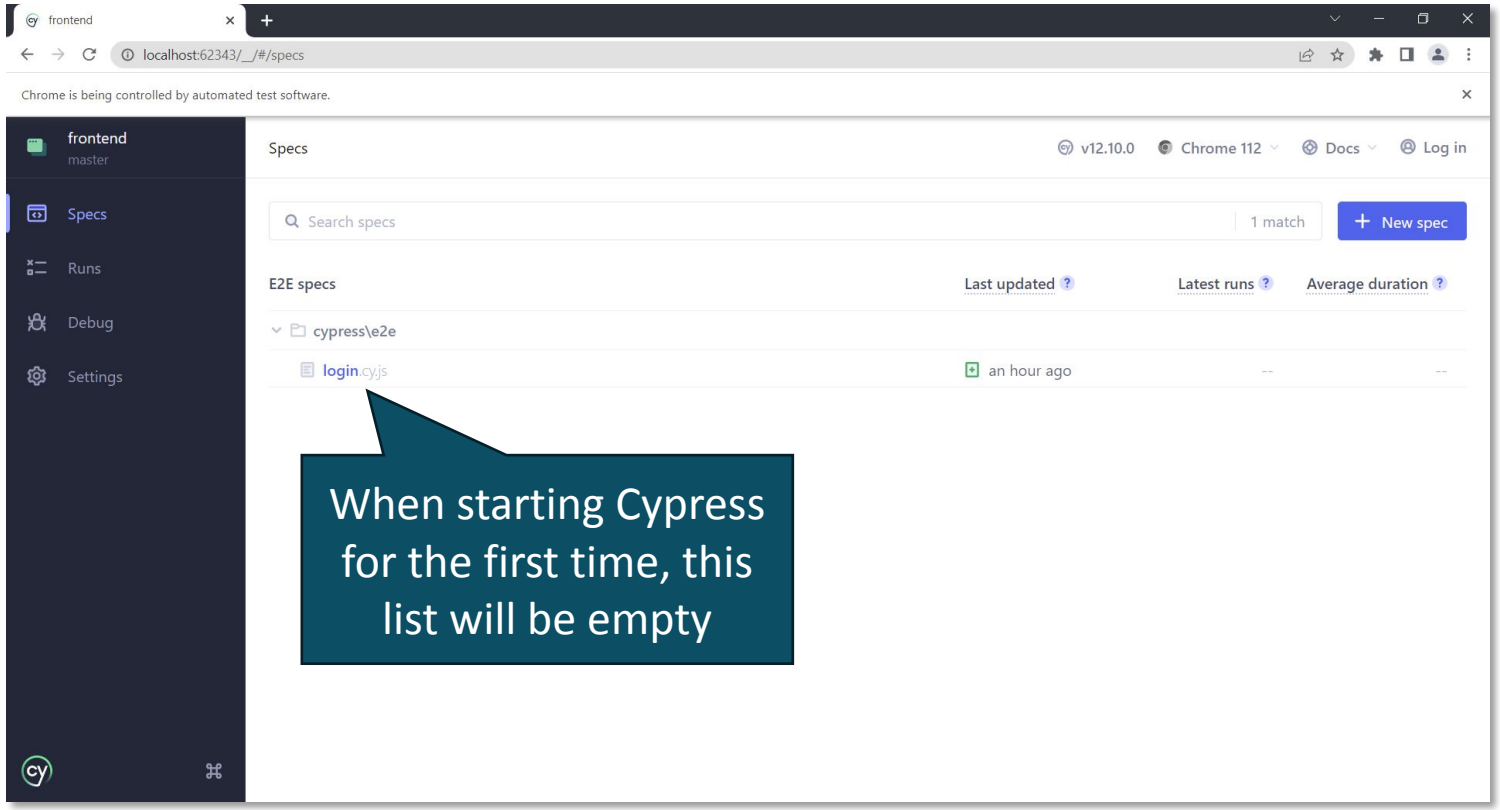
Selecting a Test Browser

BLEKINGE INSTITUTE OF TECHNOLOGY



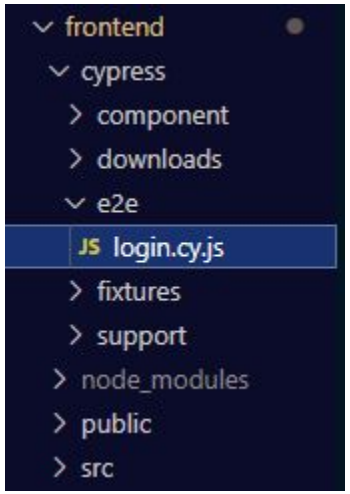
The Cypress Test Runner

BLEKINGE INSTITUTE OF TECHNOLOGY



Creating a new Spec

BLEKINGE INSTITUTE OF TECHNOLOGY




```
frontend > cypress > e2e > JS login.cy.js > ...
1   describe('Logging into the system', () => {
2     it('starting out on the landing screen', () => {
3       // enter the main main page
4       cy.visit('http://localhost:3000')
5
6       // make sure th                    header with "login"
7       cy.get('h1')
8         .should('contain.text', 'Login')
9     })
10  })
```

Name the test suite

Name a test case

Execute Cypress commands via **cy.**

Obtain a GUI element

Assert a property about the element with **should(...)**

Cypress Commands

BLEKINGE
INSTITUTE OF
TECHNOLOGY

Cypress offers a variety of commands to simulate user interaction with a website. Some of the most important being:

- **get**: get one or more DOM elements by selector or alias (similar to CSS)
- **contains**: get the DOM element containing the given text
- **find**: get the descendent DOM elements of a specific selector
- **should**: create an assertion
- **trigger**: trigger an event on a DOM element

… and many more

https://docs.cypress.io/api/table-of-contents

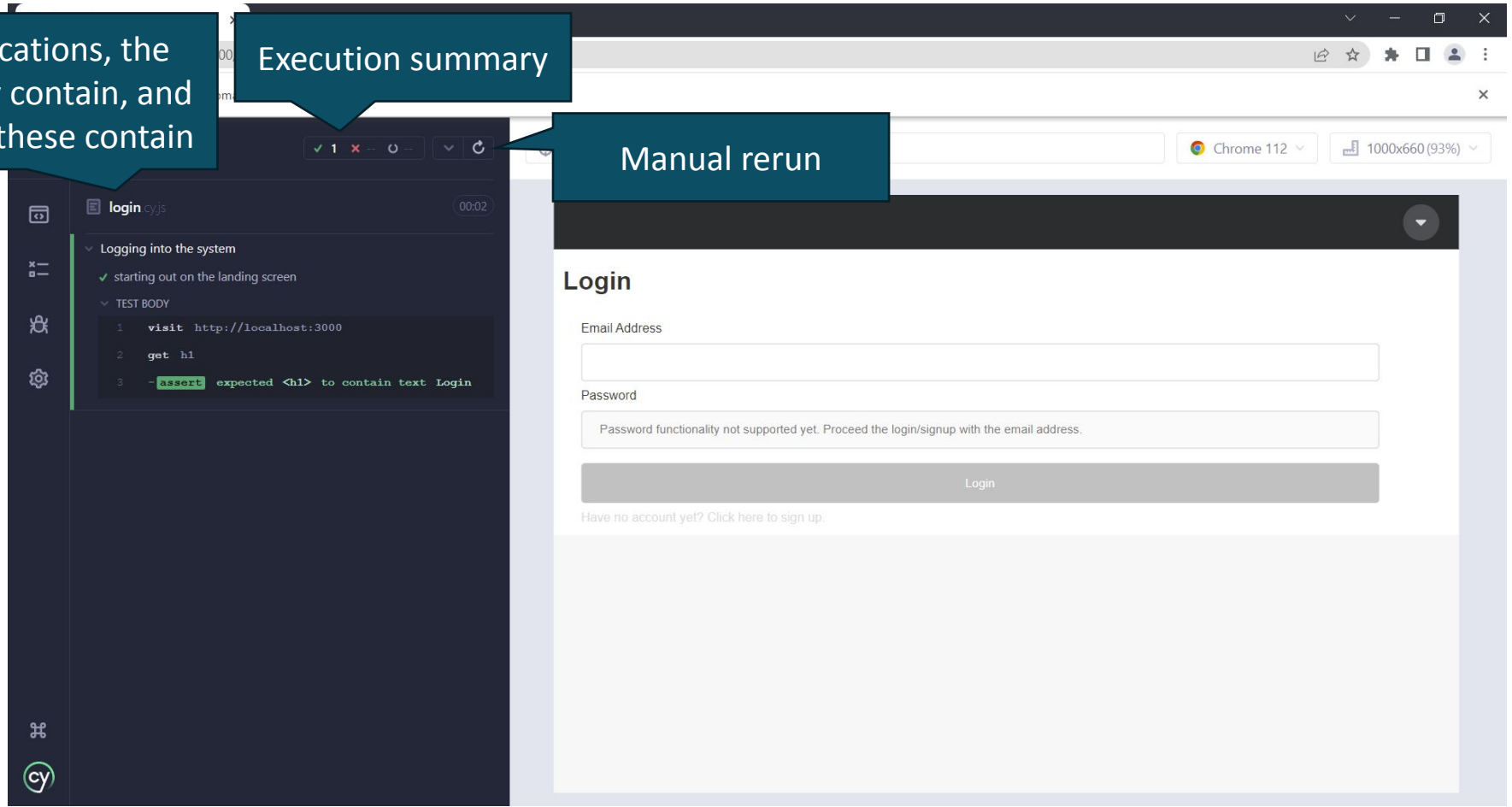
E2E Test with Cypress
Test Case Execution

BLEKINGE INSTITUTE OF TECHNOLOGY

List of specifications, the test cases they contain, and the steps that these contain

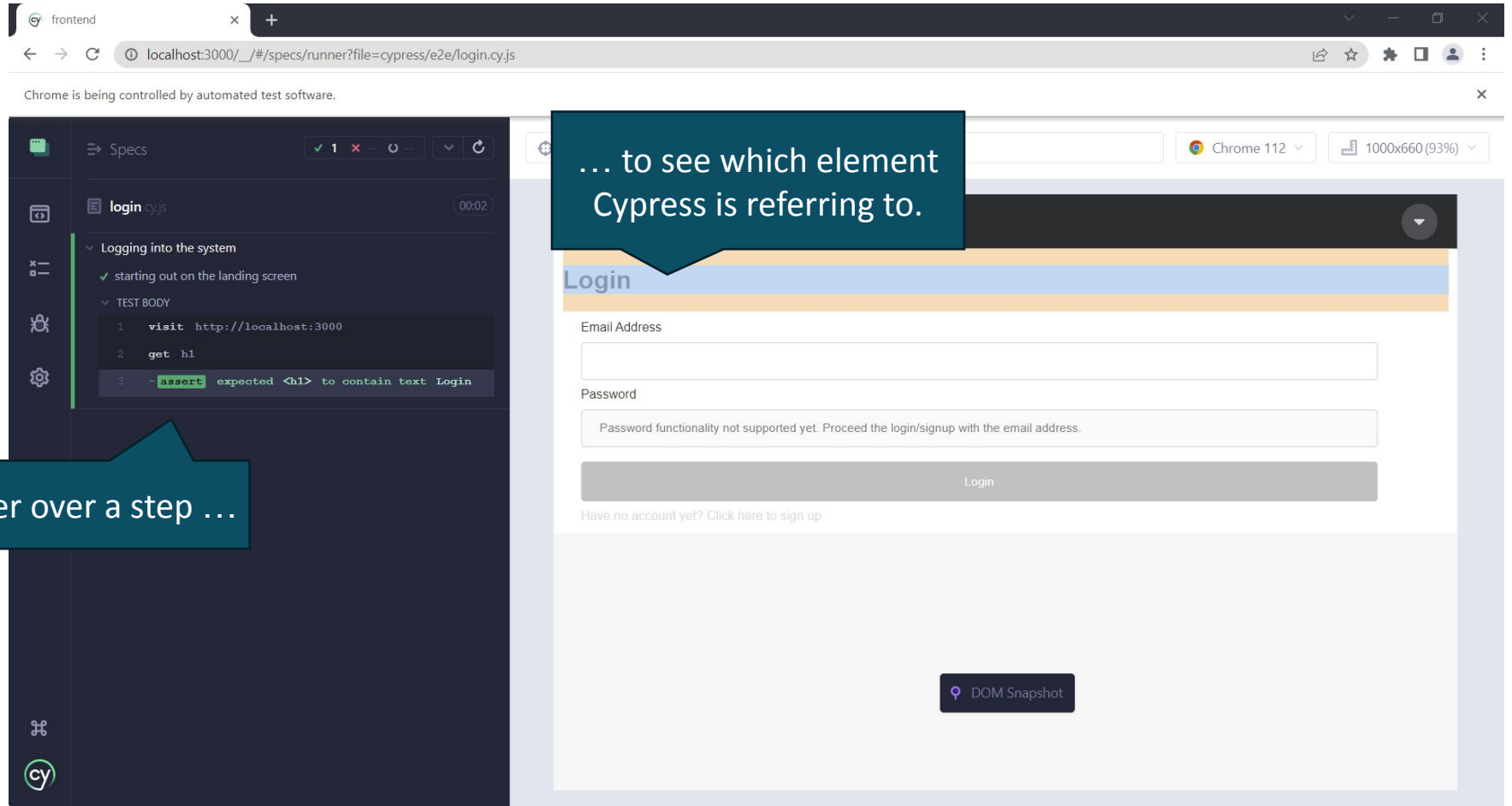
Execution summary

Manual rerun



E2E Test with Cypress
Playback

BLEKINGE INSTITUTE OF TECHNOLOGY



E2E Test with Cypress
Defining multiple Test Cases

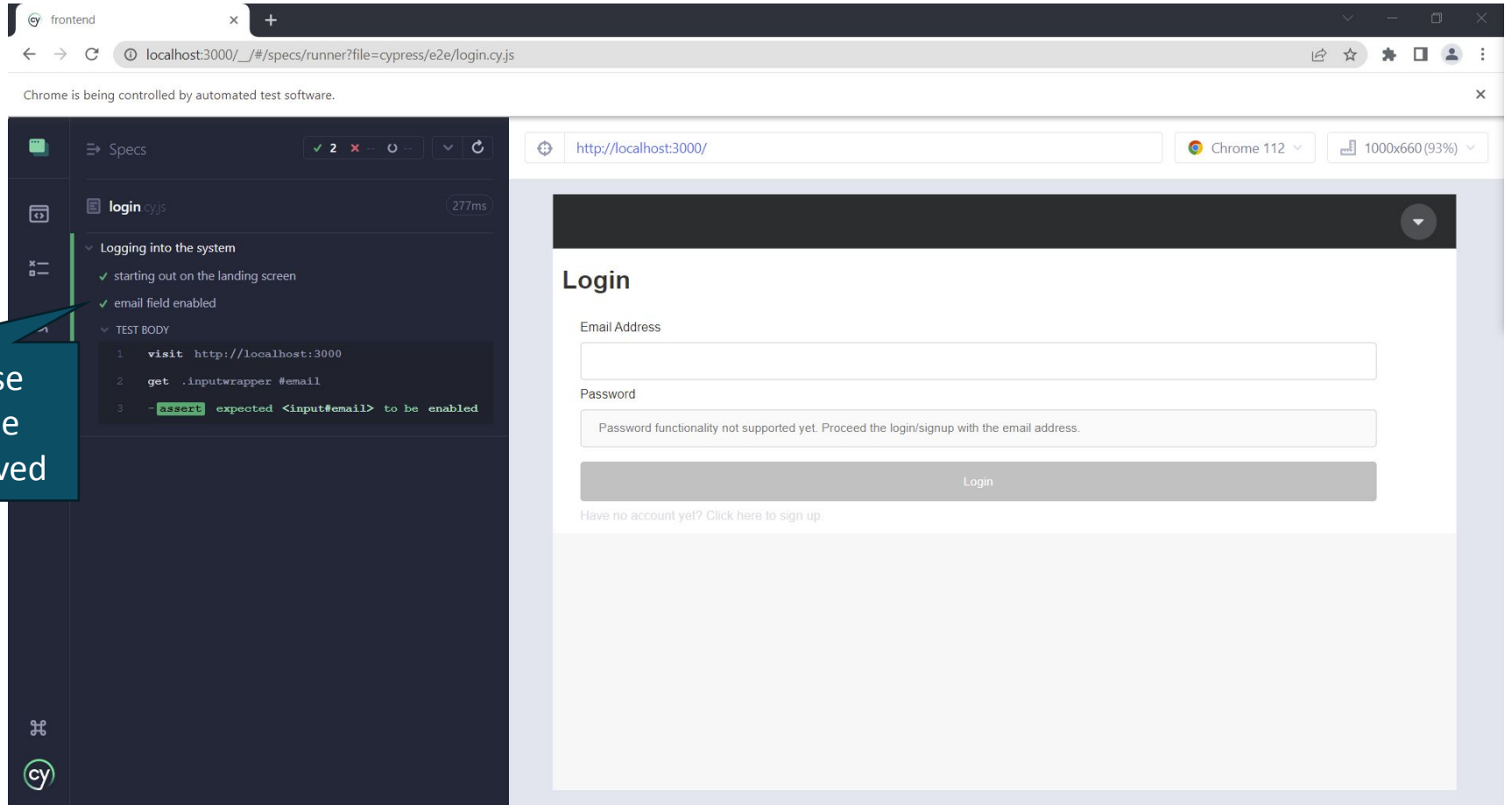
BLEKINGE INSTITUTE OF TECHNOLOGY

frontend > cypress > e2e > JS login.cy.js > ...

```
describe('Logging into the system', () => {
  it('starting out on the landing screen', () => {
    // enter the main main page
    cy.visit('http://localhost:3000')

    // make sure the landing page contains a header with "login"
    cy.get('h1')
      .should('contain.text', 'Login')
  })

  it('email field enabled', () => {
    // enter the main main page
    cy.visit('http://localhost:3000')

    // find the html element with the id email
    cy.get('.inputwrapper #email')
      .should('be.enabled')
  })
})
```

E2E Test with Cypress
Defining multiple Test Cases

BLEKINGE INSTITUTE OF TECHNOLOGY



The new test case appears once the specification is saved

E2E Test with Cypress
Test Case Setup

BLEKINGE INSTITUTE OF TECHNOLOGY

frontend > cypress > e2e > JS login.cy.js > ...

```
describe('Logging into the system', () => {
  it('starting out on the landing screen', () => {
    // enter the main main page
    cy.visit('http://localhost:3000')

    // make sure the landing page contains a header with "login"
    cy.get('h1')
      .should('contain.text', 'Login')
  })

  it('email field enabled', () => {
    // enter the main main page
    cy.visit('http://localhost:3000')

    // find the html element with the id email
    cy.get('.inputwrapper #email')
      .should('be.enabled')
  })
})
```

frontend > cypress > e2e > JS login.cy.js > ...

```
describe('Logging into the system', () => {
  beforeEach(function () {
    // enter the main main page
    cy.visit('http://localhost:3000')
  })

  it('starting out on the landing screen', () => {
    // make sure the landing page contains a header with "login"
    cy.get('h1')
      .should('contain.text', 'Login')
  })

  it('email field enabled', () => {
    // find the html element with the id email
    cy.get('.inputwrapper #email')
      .should('be.enabled')
  })
})
```

Identifying GUI Elements
Black-box vs. White-box approach

BLEKINGE INSTITUTE OF TECHNOLOGY

A significant part of Cypress test code (or any code for automatic GUI tests) is the identification of GUI/DOM elements. There are two approaches to this:

1. **Imperative** (white-box): clearly specify the element to identify based on its properties in the code
2. **Declarative** (black-box): describe how to identify certain elements the same way a user would search for them

Imperative

```
cy.get('.inputwrapper #email')
```

src/Components/LoginForm.js

```
<div className='inputwrapper'>
    <label>Email Address</label>
    <input type='text' id='email' name='email' onChange={e =>
</div>
```

Declarative

```
cy.contains('div', 'Email Address')
  .find('input[type=text]')
```

Simulate how a user would search for the respective DOM element

E2E Test with Cypress
Failing Test Cases

BLEKINGE INSTITUTE OF TECHNOLOGY

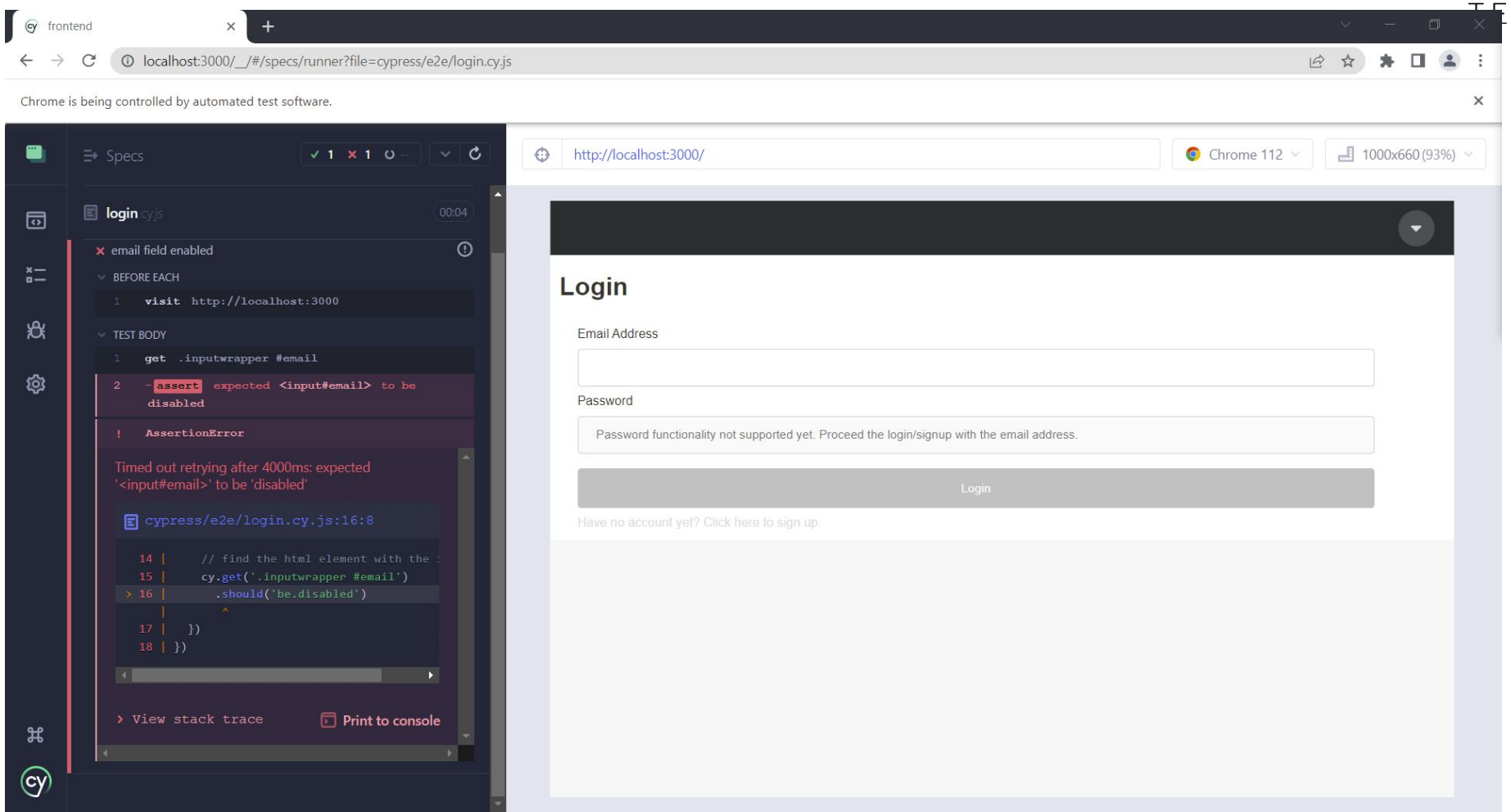
frontend > cypress > e2e > JS login.cy.js > describe('Logging into the system') callback >

```
1   describe('Logging into the system', () => {
2     beforeEach(function () {
3       // enter the main main page
4       cy.visit('http://localhost:3000')
5     })
6
7     it('starting out on the landing screen', () => {
8       // make sure the landing page contains a header with "login"
9       cy.get('h1')
10        .should('contain.text', 'Login')
11    })
12
13    it('email field enabled', () => {
14      // find the html element with the id email
15      cy.get('.inputwrapper #email')
16        .should('be.disabled')
17    })
18  })
```

This test case fails on purpose (for demonstration)

E2E Test with Cypress
Failing Test Cases

BLEKINGE INSTITUTE OF TECHNOLOGY



E2E Test with Cypress
Cypress Fixtures for Setup

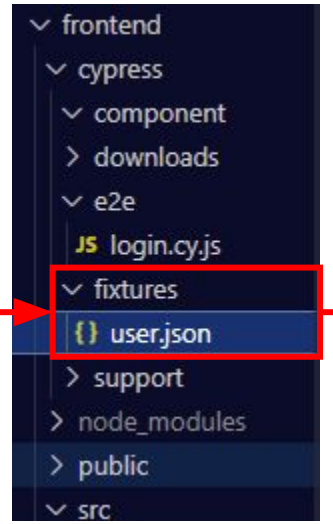
Define Variables

Issue a call to the backend

Store the returned variables

Read dummy data located in *fixtures/*

```
describe('Logging into the system', () => {
  // define variables that we need on multiple occasions
  let uid
  let name

  before(function () {
    // create a fabricated user from a fixture
    cy.fixture('user.json')
      .then((user) => {
        cy.request({
          method: 'POST',
          url: 'http://localhost:5000/users/create',
          form: true,
          body: user
        }).then((response) => {
          uid = response.body._id.$oid
          name = user.firstName + ' ' + user.lastName
        })
      })
  })

  beforeEach(function() {
    // enter the main main page
    cy.visit('http://localhost:3000')
  })
```



E2E Test with Cypress
Cypress Fixtures for Setup

BLEKINGE INSTITUTE OF TECHNOLOGY

```
describe('Logging into the system', () => {
  // define variables that we need on multiple occasions
  let uid
  let name

  before(function () {
    // create a fabricated user from a fixture
    cy.fixture('user.json')
      .then((user) => {
        cy.request({
          method: 'POST',
          url: 'http://localhost:5000/users/create',
          form: true,
          body: user
        }).then((response) => {
          uid = response.body._id.$oid
          name = user.firstName + ' ' + user.lastName
        })
      })
  })

  beforeEach(function() {
    // enter the main main page
    cy.visit('http://localhost:3000')
  })
```

Cleanup method

```
  after(function () {
    // clean up by deleting the user from the database
    cy.request({
      method: 'DELETE',
      url: `http://localhost:5000/users/${uid}`
    }).then((response) => {
      cy.log(response.body)
    })
  })
})
```

E2E Test with Cypress

```
describe('Logging into the system', () => {
  // define variables that we need on multiple occasions
  let uid
  let name

  before(function () {
    // create a fabricated user from a fixture
    cy.fixture('user.json')
      .then((user) => {
        cy.request({
          method: 'POST',
          url: 'http://localhost:5000/users/create',
          form: true,
          body: user
        }).then((response) => {
          uid = response.body._id.$oid
          name = user.firstName + ' ' + user.lastName
        })
      })
  })

  beforeEach(function() {
    // enter the main main page
    cy.visit('http://localhost:3000')
  })
```

```
  after(function () {
    // clean up by deleting the user from the database
    cy.request({
      method: 'DELETE',
      url: `http://localhost:5000/users/${uid}`
    }).then((response) => {
      cy.log(response.body)
    })
  })
})
```

```
  it('starting out on the landing screen', () => {
    // make sure the landing page contains a header with "login"
    cy.get('h1')
      .should('contain.text', 'Login')
  })

  it('login to the system with an existing account', () => {
    // detect a div which contains "Email Address", find the input and type
    // declarative
    cy.contains('div', 'Email Address')
      .find('input[type=text]')
      .type('mon.doe@gmail.com')
    // alternative, imperative way of detecting that input field
    //cy.get('.inputwrapper #email')
    //    .type('mon.doe@gmail.com')

    // submit the form on this page
    cy.get('form')
      .submit()

    // assert that the user is now logged in
    cy.get('h1')
      .should('contain.text', 'Your tasks, ' + name)
  })
```

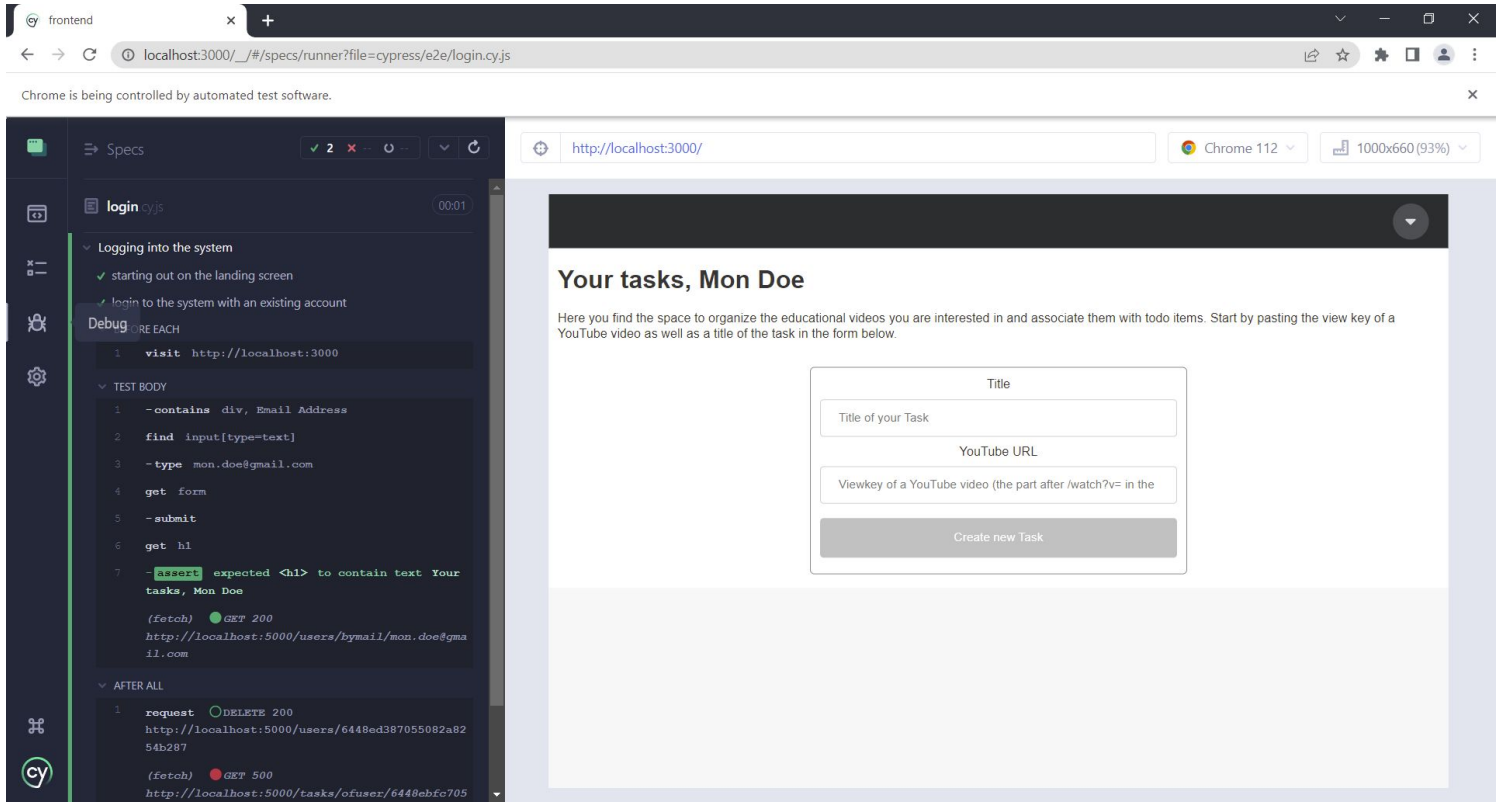
Find the input form and enter the email address of the dummy user

Submit the input form

Assert that the user is now logged in

E2E Test with Cypress

BLEKINGE INSTITUTE OF TECHNOLOGY



GUI Testing Approach Summary

BLEKINGE
INSTITUTE OF
TECHNOLOGY

Our approach to testing has not changed:

1. **Test design**: transform a use case into one or more test cases (using the 4-step test design technique)
2. **Test implementation**: translate each test case into Cypress code
3. **Test execution and evaluation**: Run the tests and evaluate the test result

All that has changed is:

- **The ground truth/oracle**: now we use use cases instead of docstrings
- **The testing framework**: now we use Cypress instead of Pytest


BLEKINGE TEKNISKA HÖGSKOLA
BTH