

Agenda

1. Selenium WebDriver

- a. wprowadzenie
- b. demo
- c. zadania

2. Splinter

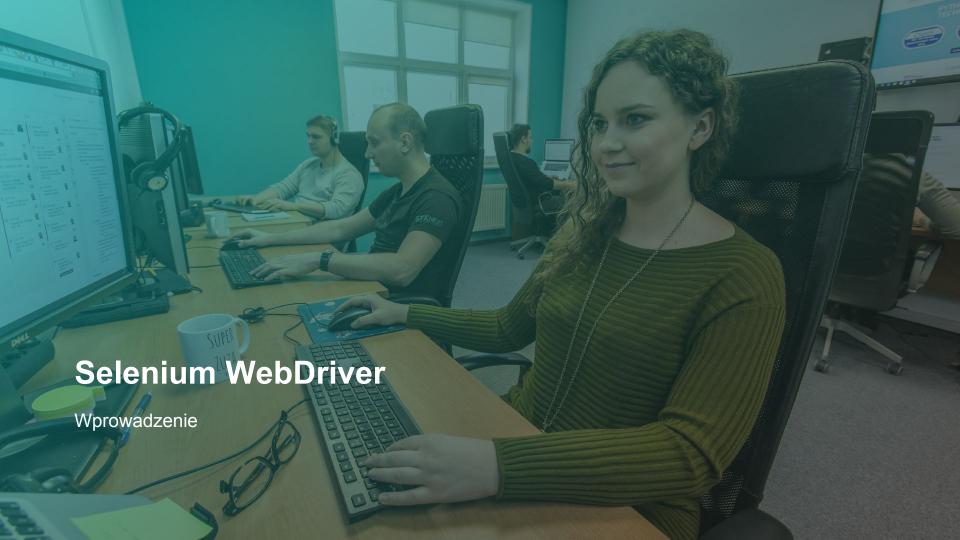
- a. wprowadzenie
- b. demo
- c. zadania

3. Unittest

- a. wprowadzenie
- b. demo
- c. zadania

4. Behave

- a. wprowadzenie
- b. demo
- c. zadania



Selenium WebDriver

Interfejs do komunikacji z przeglądarką internetową umożliwiający w pełni zautomatyzowaną interakcję z wszystkimi elementami wyświetlanych stron.

>> pip install selenium

Przykład

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
browser = webdriver.Firefox()
browser.get("http://www.python.org")
assert "Python" in browser.title
elem = browser.find_element_by_name("q")
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No results found." not in browser.page_source
browser.close()
```

Podstawowe metody

```
# Instancja przeglądarki
browser = webdriver.Firefox()
browser.quit()
browser.refresh()
# Nawigacja
browser.get('https://stxnext.com')
browser.back()
browser.forward()
# Treść
browser.title
browser.current_url
browser.page_source
```

Szukanie

```
# Szukanie elementów
browser.find_element_by_css_selector('h1')
browser.find_element_by_xpath('//h1')
browser.find_element_by_tag_name('h1')
browser.find_element_by_name('name')
browser.find_element_by_class_name('name')
browser.find_element_by_id('firstheader')
browser.find element by value('query')
# Szukanie linków
browser.find_element_by_link_text(
      'Link for Example.com'
browser.find_element_by_partial__link_text(
      'for Example'
```

```
# Gdy elementów jest wiele
browser.find_elements_by_*

# Szukanie wewnątrz komponentu
div = browser.find_element_by_tag_name("div")
div.find_element_by_name("name")
```

Elementy

```
# Właściwości
element.id
element.tag name
element.text
element.is_displayed()
element.is_enabled()
element.is selected()
element.get property(name)
element.get attribute(name)
element.value_of_css_property(name)
```

```
# Interakcje
element.click()
element.clear()
element.submit()
element.find element*
element.send_keys(keys)
```

Formularze

```
# Select
# Pola tekstowe
field = browser.find_element_by_name(
                                                    from selenium.webdriver.support.ui import Select
     'fullName'
                                                    select = Select(
field.clear()
                                                         browser.find element by name('name')
field.send keys('Full Name')
# Button, checkbox, radio
                                                    select.select by index(index)
btn = browser.find element by name('submit')
                                                    select.select by visible text('text')
btn.click()
                                                    select.select by value(value)
# Wysłanie formularza
                                                    select.deselect by *
field.submit()
                                                    select.deselect all()
```

Oczekiwanie na zmiany

```
# Oczekiwanie na spełnienie warunku
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected conditions as EC
wait = WebDriverWait(browser, 10)
wait.until(browser, EC.presence of element located((By.ID, 'my id'))
# Ustawienie globalne
browser.implicitly wait(10)
```

Oczekiwanie na zmiany - warunki

```
# Warunki
EC.title is(text)
EC.title contains(text)
EC.presence of element_located(locator)
EC.presence of all elements located(locator)
EC.visibility of element located(locator)
EC.invisibility of element located(locator)
EC.element located to be selected(locator)
EC.element located selection state to be(locator, is selected)
EC.element to be clickable(locator)
EC.element to be selected(element)
EC.element selection state to be(element, is selected)
EC.staleness of(element)
EC.text to be present in element(locator, text)
```

```
# Typy lokatorów

By.ID

By.XPATH

By.LINK_TEXT

By.PARTIAL_LINK_TEXT

By.NAME

By.TAG_NAME

By.CLASS_NAME

By.CSS SELECTOR
```

Inne

- obsługa wielu okien przeglądarki
- obsługa ruchów myszy, drag and drop, double click, right click
- wykonywanie kodu javascript
- cookies injection

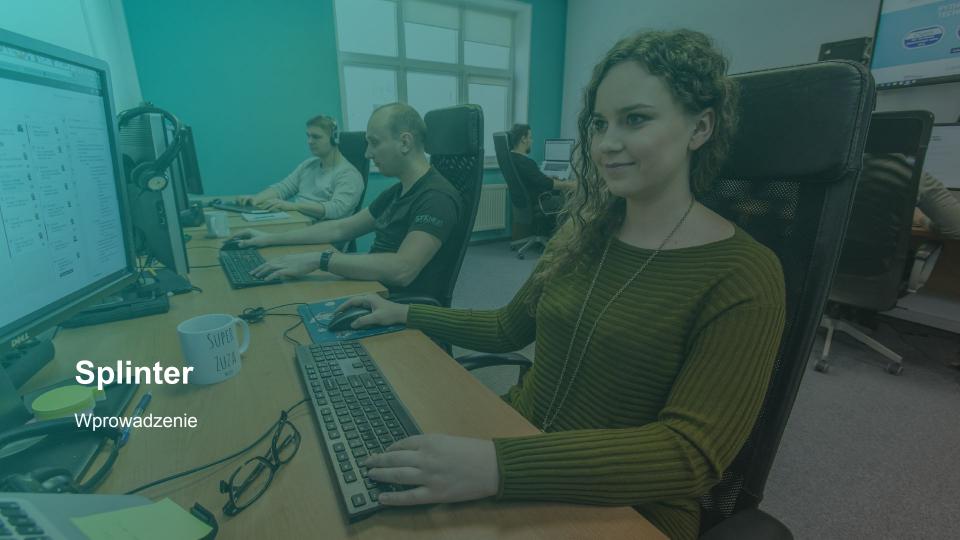




Zadania cz I: Selenium WebDriver

- 1. Używając konsoli Pythonowej i biblioteki selenium
 - a. przejdź do strony https://duckduckgo.com/
 - b. wyszukaj frazę "the biggest python software house"
 - c. przejdź do pierwszego wyniku na liście
 - d. sprawdź czy gdzieś na stronie występuje tekst "STX Next"

- Używając konsoli Pythonowej i biblioteki selenium
 - a. zaloguj się do aplikacji DiabControl
 - b. przejdź do strony ze wszystkimi pacjentami
 - c. wydrukuj do konsoli pacjentów
 - i. same adresy email
 - ii. dane pacjenta w formacie "<lmię> <Nazwisko> (<email>)"



Czym jest Splinter?

- wysokopoziomowa biblioteka dla Pythona
- jest używany jako adapter do Selenium WebAPI
- usprawnia pracę na obiektach
- poprawia czytelność kodu
- kod napisany raz działa na różnych przeglądarkach

>> pip install splinter

Przykład

```
from splinter import Browser
with Browser() as browser:
   # Visit URL
   browser.visit("http://www.google.com")
   browser.fill('q', 'splinter - python acceptance testing for web applications')
   # Find and click the 'search' button
   button = browser.find by name('btnG')
   # Interact with elements
   button.click()
   if browser.is_text_present('splinter.readthedocs.io'):
       print("Yes, the official website was found!")
   else:
       print("No, it wasn't found... We need to improve our SEO techniques")
```

Podstawowe metody

```
# Instancja przeglądarki
browser = Browser()
browser.quit()
browser.reload()
# Nawigacja
browser.visit('https://stxnext.com')
browser.back()
browser.forward()
# Treść
browser.title
browser.url
browser.html
```

Formularze

Pola są identyfikowane przez atrybut "name".

```
<form>
 Name: <input type="text" name="fullName"><br>
 Email: <input type="text" name="email"><br>
</form>
browser.fill('fullName', 'my name')
browser.attach_file('file', '/path/to/file/somefile.jpg')
browser.choose('some-radio', 'radio-value')
browser.check('some-check')
browser.uncheck('some-check')
browser.select('uf', 'rj')
```

```
# Przyciski
btn = browser.find_by_name('submit')
btn.click()
```

Szukanie

```
# Szukanie elementów
browser.find_by_css('h1')
browser.find_by_xpath('//h1')
browser.find_by_tag('h1')
browser.find by name('name')
browser.find_by_text('Hello World!')
browser.find by id('firstheader')
browser.find by value('query')
# Szukanie linków
browser.find_link_by_text('Link for Example.com')
browser.find_link_by_partial_text('for Example')
browser.find link by href('http://example.com')
browser.find link by partial href('example')
```

```
# Gdy elementów jest wiele
browser.find_by_name('name').first
browser.find_by_name('name').last
browser.find by name('name')[1]
# Szukanie wewnątrz komponentu
divs = browser.find by tag("div")
divs.first.find_by_name("name")
```

Dostępność tekstu

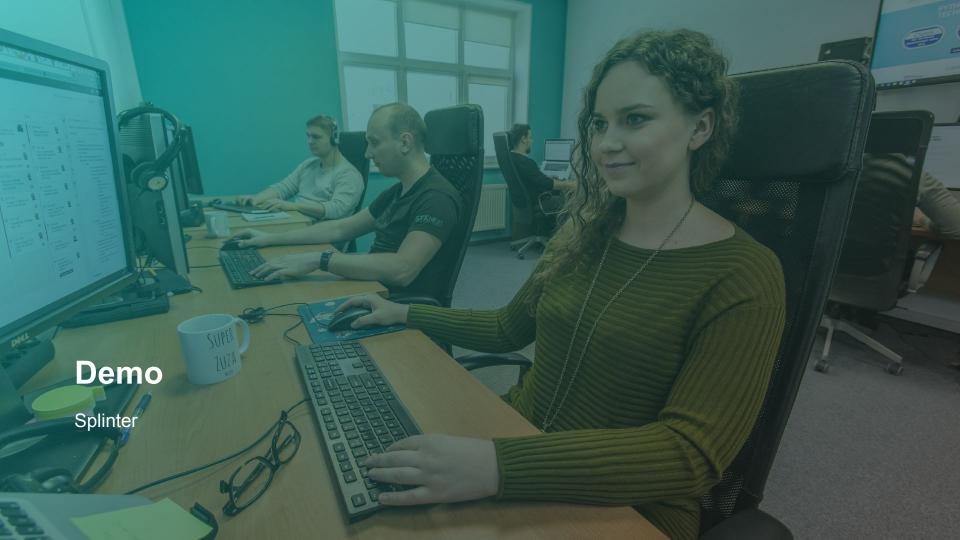
```
browser.is_text_present("Done")
browser.is_text_present('splinter', wait_time=10)
browser.is_text_not_present('text not present')
browser.is_text_not_present('text not present', wait_time=10)
```

Dostępność elementów

```
browser.is_element_not_present_by_css('h6')
browser.is_element_present_by_css('h1')
browser.is_element_present_by_xpath('//h1')
                                                   browser.is_element_not_present_by_xpath('//h6')
browser.is_element_present_by_tag('h1')
                                                   browser.is_element_not_present_by_tag('h6')
browser.is_element_present_by_name('name')
                                                   browser.is_element_not_present_by_name('unexisting')
browser.is_element_present_by_text('Hello!')
                                                   browser.is_element_not_present_by_text('Not here :(')
browser.is_element_present_by_id('firstheader')
                                                   browser.is_element_not_present_by_id('unexisting-head')
browser.is element present by value('query')
                                                   browser.is element not present by id(
browser.is_element_present_by_value(
                                                          'unexisting-head',
      'query',
                                                         wait time=10
     wait time=10
```

Inne

- obsługa wielu okien przeglądarki
- obsługa ruchów myszy, drag and drop, double click, right click
- wykonywanie kodu javascript
- cookies injection

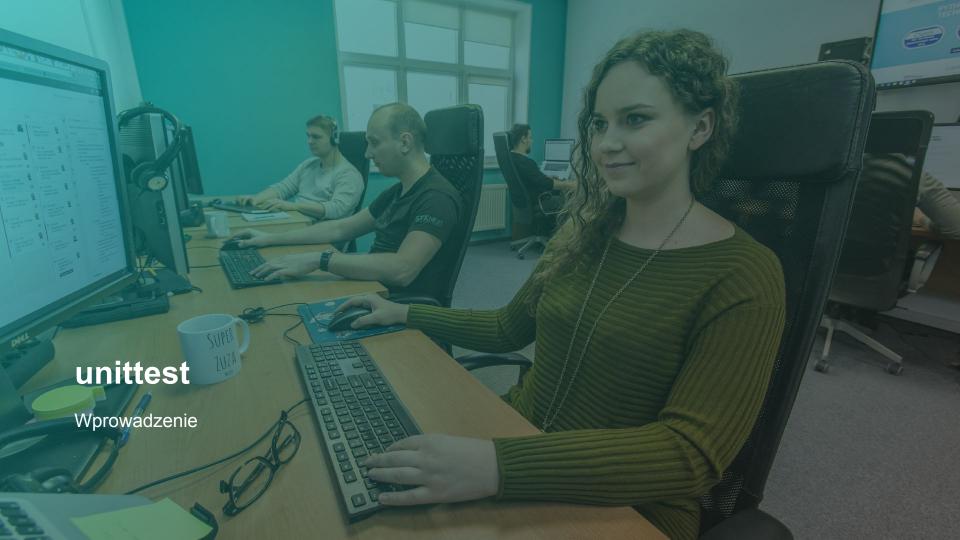




Zadania cz II: Splinter

- 1. Używając konsoli Pythonowej i biblioteki Splinter
 - a. przejdź do strony https://duckduckgo.com/
 - b. wyszukaj frazę "the biggest python software house"
 - c. przejdź do pierwszego wyniku na liście
 - d. sprawdź czy gdzieś na stronie występuje tekst "STX Next"

- 2. Używając konsoli Pythonowej i biblioteki Splinter
 - a. zaloguj się do aplikacji DiabControl
 - b. przejdź do strony ze wszystkimi pacjentami
 - c. wydrukuj do konsoli pacjentów
 - i. same adresy email
 - ii. dane pacjenta w formacie "<lmię> <Nazwisko> (<email>)"



Czym są unittesty

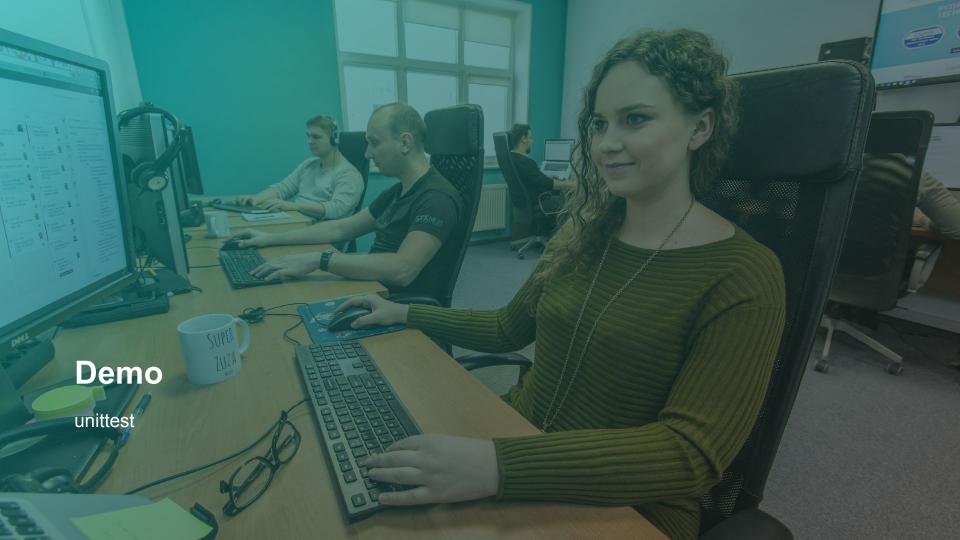
- testy weryfikujące poprawność działania pojedynczych elementów
- porównanie otrzymanych wyników z wynikami oczekiwanymi
- łatwość w zarządzaniu kodem

```
Przykład
import unittest
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
class PythonOrgSearch(unittest.TestCase):
   def setUp(self):
       self.browser = webdriver.Firefox()
   def tearDown(self):
       self.browser.quit()
   def test_search_in_python_org(self):
       browser = self.browser
       browser.get("http://www.python.org")
       self.assertIn("Python", browser.title)
       elem = browser.find_element_by_name("q")
       elem.send_keys("pycon")
       elem.send_keys(Keys.RETURN)
       assert "No results found." not in browser.page_source
if __name__ == "__main__":
```

unittest.main()

Asercje

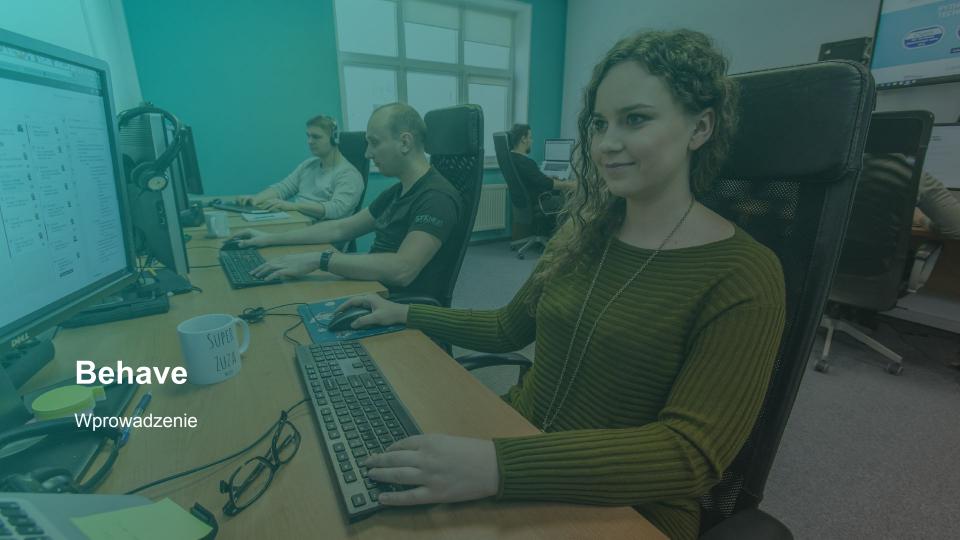
```
button = browser.find_element_by_class_name('my-button')
self.assertEqual(button.text, 'Submit')
headers = browser.find_elements by tag name('th')
headers text = [header.text for header in headers]
self.assertIn('First Name', headers text)
expected headers = ['Email', 'First Name', 'Last name']
self.assertListEqual(expected headers, headers text)
hidden element = browser.find element by id('hide-me')
self.assertFalse(hidden element.is displayed())
checkbox = browser.find_element_by_value('checkbox_one')
self.assertTrue(checkbox.is selected())
```





Zadania cz III: unittest

- 1. Stwórz plik z TestCasem na podstawie przykładu ze slajdu 30
 - a. stwórz nową instancję przeglądarki przed każdym testem
 - b. zamknij instancję przeglądarki po każdym teście
- 2. Zaimplementuj test sprawdzający, że po zalogowaniu do aplikacji DiabControl wyświetlany jest tekst 'Welcome to DiabControl system'
 - a. przejdź na stronę aplikacji
 - b. zaloguj się
 - c. sprawdź czy podany tekst się wyświetla
- 3. Zaimplementuj test sprawdzający, że w zakładce 'My patients' wyświetla się tabela z nagłówkami 'Email', 'First Name', 'Last Name'
 - a. przejdź na stronę aplikacji
 - b. zaloguj się jako doktor
 - c. kliknij w link 'My patients'
 - d. sprawdź czy nagłówki w tabeli są poprawne



Czym jest Behave

- wysokopoziomowa biblioteka dla Pythona
- tłumacz składni Gherkina na metody w Pythonie
- technika Behaviour Driven Development

>> pip install behave

Przykład

```
Feature: showing off behave

Scenario: run a simple test

Given we have behave installed

When we implement a test

Then behave will test it for us!
```

```
steps/simple_test.py
from behave import *
@given('we have behave installed')
def step_impl(context):
   pass
@when('we implement a test')
def step_impl(context):
   assert True is not False
@then('behave will test it for us!')
def step_impl(context):
   assert context.failed is False
```

Struktura katalogów

Przykład 1

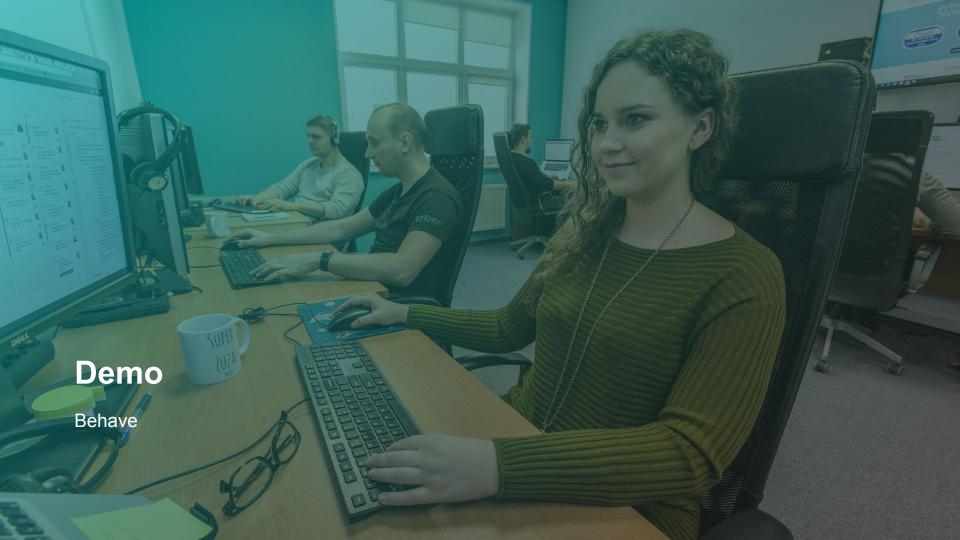
features/
features/environment.py
features/everything.feature
features/steps/
features/steps/steps.py

Przykład 2

```
features/
features/signup.feature
features/login.feature
features/account_details.feature
features/environment.py
features/steps/
features/steps/website.py
features/steps/utils.py
```

environments.py

```
def before_step(context, step):
                                                            def after_step(context, step):
   pass
                                                                pass
def before_scenario(context, scenario):
                                                            def after_scenario(context, scenario):
   pass
                                                                pass
def before_feature(context, feature):
                                                            def after_feature(context, feature):
   pass
                                                                pass
def before_tag(context, tag):
                                                            def after_tag(context, tag):
   pass
                                                                pass
                                                            def after_all(context):
def before_all(context)
   pass
                                                                pass
```





Zadania cz IV: Behave

- 1. Przygotuj strukturę projektu
 - utwórz odpowiednie pliki i katalogi zgodnie z przykładem nr 1, slajd 38
 - utwórz instancję przeglądarki przed uruchomieniem scenariusza
 - zamknij przeglądarkę po zakończeniu
 - d. uruchom test

2. Zaimplementuj poniższy scenariusz

Feature: Registration

Scenario: Registration passed

Given I have access to the system

And I am on registration page

When I submit the form with valid patient details

Then the registration passed

Scenario: Registration failed

Given I have access to the system

And I am on registration page

When I submit the form with invalid patient details

Then the registration failed

Zadania cz IV: Behave

3. Zaimplementuj poniższy scenariusz

Feature: Chat

```
Scenario: Sending the message

Given I am logged in as doctor

And I have a patient assigned to me

And I am on patent card details page

When I type the message in chat window

Then the message has been sent
```

Prelegenci:

Rafał Nowicki rafal.nowicki@stxnext.pl

Paweł Ostrowski pawel.ostrowski@stxnext.pl

QUESTIONS

Rekrutacja:

Oleksandra Chernyak oleksandra.chernyak@stxnext.pl

Joanna Trubisz joanna.trubisz@stxnext.pl

@STXNext

f in





