Automatyzacja przypadków testowych przy pomocy Selenium Webdriver z zastosowaniem Page Object Pattern.

I. Przypadki testowe logowania z poprawnymi i nieprawidłowymi danymi

ID: 001
Tytuł: Logowanie do systemu z prawidłowymi danymi
Warunek wstępny: Uruchomiona przeglądarka. Użytkownik nie jest zalogowany. Użytkownik posiada konto w systemie.

Kroki:
1. Wejdź na stronę http://demowebshop.tricentis.com/
2. Kliknij link Log-in
3. Wpisz prawidłowy email
4. Wpisz prawidłowe hasło
5. Kliknij przycisk Login-in

Oczekiwany rezultat:
Użytkownikowi udaje się zalogować do systemu.
W górnym menu nazwa konta jest taka sama jak wprowadzonego emaila.

ID: 002
Tytuł: Logowanie do systemu z nieprawidłowym email
Warunek wstępny: Uruchomiona przeglądarka. Użytkownik nie jest zalogowany.

Kroki:
1. Wejdź na stronę http://demowebshop.tricentis.com/
2. Kliknij link Log-in
3. Wpisz niepoprawny email
4. Wpisz prawidłowe hasło
5. Kliknij przycisk Login-in

Oczekiwany rezultat:
Użytkownikowi nie udaje się zalogować do systemu.
Wyświetla się error message: "Please enter a valid email address."

II. Automatyzacja przypadków testowych przy pomocy Selenium Webdriver

Przypadki testowe znajdują się w pliku LoginTest.py w klasie LoginTest

```
from selenium import webdriver
import unittest
from selenium.webdriver.common.keys import Keys
import sys
import os
sys.path.append(os.path.join(os.path.dirname(__file__), "..", ".."))
from Logintests.Tests.BaseTest import BaseTest
from Logintests.Pages.header import Header
from Logintests.Pages.loginPage import LoginPage

valid_email = "tadeusz.kaminski21cn@gmail.com"
invalid_email = "abcd"
valid_password = "tester1"

class LoginTest(BaseTest):

    def test_valid_login(self):
        header = Header(self.driver)
        header.click_login_link()
        login = LoginPage(self.driver)
        login.enter_email(valid_email)
        login.enter_password(valid_password)
        login.click_login_button()
         self.assertEqual(header.get_user_account_name(), valid_email)

    def test_login_with_invalid_email(self):
        header = Header(self.driver)
        header.click_login_link()
        login = LoginPage(self.driver)
        login.enter_email(invalid_email)
        login.enter_password(valid_password)
        login.click_login_button()
        message = "Please enter a valid email address."
        self.assertEqual(login.get_email_error_message(), message)


  if __name__ == '__main__' :
      unittest.main()
```

Klasa LoginTest dziedziczy po klasie **BaseTest**, która zawiera metodę **setUp,** która jest wykonywana przed każdym testem, oraz metodę **tearDown**, która jest wykonywana po zakończeniu każdego testu.

```
import unittest
from selenium import webdriver
import sys
import os
sys.path.append(os.path.join(os.path.dirname(__file__), "..", ".."))

class BaseTest(unittest.TestCase):

   def setUp(self):
      self.driver = webdriver.Chrome('../chromedriver.exe')
      driver = self.driver
      driver.get('http://demowebshop.tricentis.com/')
      driver.maximize_window()
      driver.implicitly_wait(10)

  def tearDown(self):
     self.driver.quit()

if  __name__ == '__main__' :
   unittest.main()
```

Dla strony logowania utworzono klasę **LoginPage**, która zawiera elementy znajdujące się na stronie logowania, oraz metody obsługujące te elementy.

```
class LoginPage:

    def __init__(self, driver):
       self.driver = driver
       self.email_textbox_id = "Email"
       self.password_textbox_id = "Password"
       self.login_button_class = "login-button"
       self.email_validation_messagge_xpath = "//span[@for ='Email']"


   def enter_email(self, email):
      self.driver.find_element_by_id(self.email_textbox_id).clear()
      self.driver.find_element_by_id(self.email_textbox_id).send_keys(email)
```

```
 def enter_password(self, password):
     self.driver.find_element_by_id(self.password_textbox_id).clear()
     self.driver.find_element_by_id(self.password_textbox_id).
     send_keys(password)

 def click_login_button(self):
     self.driver.find_element_by_class_name(self.login_button_class).click()

 def get_email_error_message(self):
     message = self.driver.find_element_by_xpath
     (self.email_validation_messagge_xpath).text

 return message
```

Dla elementów headera utworzono osobną klasę **Header**

```
class Header:

  def __init__(self, driver):
      self.driver = driver
      self.login_link_xpath ="//div[contains(@class,
      'header')]//a[contains(@class, 'ico-login')]"
      self.customer_data_link_xpath = "//div[contains(@class,
      'header')]//a[contains(@class, 'account')]"
      self.log_out_link_xpath = "//div[contains(@class,
      'header')]//a[contains(@class, 'ico-logout')]"

 def click_login_link(self):
      self.driver.find_element_by_xpath(self.login_link_xpath).click()

 def get_user_account_name(self):
      user_data_link =
      self.driver.find_element_by_xpath(self.customer_data_link_xpath)
      user_data = user_data_link.text
      return user_data

 def logout(self):
    self.driver.find_element_by_xpath(self.log_out_link_xpath).click()
```

III. Uwagi końcowe

Automatyzacja przypadków testowych powiodła.
Ze względu na zastosowanie wzorca Page Object Pattern, jeżeli lokatory elementów na stronie ulegną modyfikacji wystarczy zmienić je w odpowiednich klasach (Header, LoginTest).
Same testy są odporne na zmianę struktury strony.

Dodatkowe uwagi dotyczące środowiska zostały umieszczone w pliku README.md

Projekt znajduej się na githubie pod adresem:
https://github.com/testertadek/Logintests.git