НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Курсова робота

із дисципліни «Автоматизоване тестування програмного забезпечення»

на тему

Тестування веб-застосунку за допомогою

Python та Selenium WebDriver

|  |  |
| --- | --- |
| Виконала: | Керівник: |
| студентка групи КМ-81 | *ст. викладач Бай Ю. П.* |
| *Приставська М.М* |  |

Київ — 2020

**Зміст**

[Мета роботи 2](#_Toc60089783)

[1. Постановка завдання 3](#_Toc60089784)

[2. Специфікація 6](#_Toc60089785)

[Висновки 8](#_Toc60089786)

[Список літератури 9](#_Toc60089787)

[Додаток 1. Відповіді на контрольні питання 10](#_Toc60089788)

[Додаток 2. Текст програми тестування 11](#_Toc60089789)

[Додаток 6. Скріншоти результатів виконання тестів 20](#_Toc60089790)

# Мета роботи

Метою курсової роботи є створення автоматизоване тестування локального веб-застосунку за допомогою Python та Selenium WebDriver, бібліотеки unittest.

Selenium WebDriver, або просто WebDriver - це драйвер браузера, тобто не має призначеного для користувача інтерфейсу програмна бібліотека, яка дозволяє різним іншим програмам взаємодіяти з браузером, управляти його поведінкою, отримувати від браузера якісь дані і змушувати браузер виконувати якісь команди.

Результатом виконання роботи є ознайомлення з основними можливостями Selenium WebDriver та створення готового програмного продукту для повного покриття тестами функціоналу веб-застосунку згідно специфікації.

# Постановка завдання

Створити тестовий модуль — скрипт на мові Python, який в автоматичному режимі запускає веб-сервер та виконує наступні дії:

1. Перевіряє, що анонімний користувач бачить на сторінці:
   1. Заголовок питання, який співпадає з очікуваним заголовком.
   2. Текст питання (переконатися, що блок під заголовком є непустим).
   3. Коментарі до питання, які існують за замовчуванням (переконатися, що співпадають тексти та відповідні автори коментарів).
   4. Кнопки **Sign Up**, **Log In**.
   5. Поля для введення ***Display Name, Email*** *та* ***Password*** та їх підписи.
2. Перевіряє, що новий користувач може зареєструватись, увівши значення *Display Name, Email* та *Password*:
   1. Протестувати, що у випадку успішної реєстрації користувач автоматично входить у свій обліковий запис, на сторінці з’являється його *Display Name* та кнопка **Log Out.**
   2. Протестувати, що натискання на кнопку **Log Out** призводить до виходу користувача зі свого облікового запису
   3. Протестувати, що при неунікальному значенні *Email* на *backend* створення нового користувача не відбувається, веб-застосунок не переходить в стан зареєстрованого користувача.
3. Перевіряє, що зареєстрований користувач, увівши правильні значення *Email* та *Password* та натиснувши на кнопку **Log In,** входить до свого облікового запису. Протестувати, що при введенні некоректного *Email* або *Password —* користувач не входить до свого облікового запису.
4. Перевіряє, що якщо користувач знаходиться у своєму обліковому запису, то на сторінці є **Редактор** для введення нового коментаря та кнопка **New comment**. Виконати JS script, що робить border елемента з id = “editor section” червоного кольору. Зробити скріншоти веб-сторінки засобами Selenium WebDriver.
5. Перевіряє, що користувач, який знаходиться у своєму обліковому запису, може ввести новий коментар:
   1. Після введення тексту коментаря та натискання на кнопку **New comment** створений коментар відображається у списку всіх коментарів.
   2. Текст коментаря, відформатований в Редакторі за допомогою кнопок **B**, *I*, U, ~~O~~ відображається у коментарі так, як очікується.
   3. Поруч зі створеним коментарем присутнє ім’я користувача *Display Name* та кнопка **Remove**.
6. Перевіряє, що користувач, який знаходиться у своєму обліковому запису, може ввести ще один коментар:
   1. Переконатися, що вдалося створити ще один коментар.
   2. Переконатися, що коментар знаходиться після попереднього коментаря, створеного даним користувачем.
7. Перевіряє, що користувач, який знаходиться у своєму обліковому запису, може видалити створений ним коментар:
   1. Переконатися, що видаляється потрібний коментар.
   2. Переконатися, що усі інші коментарі, створені даним користувачем та іншими користувачами, залишилися на місці.
8. Перевіряє, що одночасно у своїх облікових записах можуть бути активними два різних користувача:
   1. Увійти в браузері *driver1* в обліковий запис користувача *user1* та створити коментар *comment1*.
   2. В автоматичному режимі запустити ще один браузер *driver2*, виконати **Log In** від імені користувача *user2* та створити коментар *comment2*. Протестувати, що в браузері *driver2* присутні обидва коментаря — *comment1* та *comment2*.
   3. В браузері *driver1* від імені користувача *user1* додати коментар *comment3*. Протестувати, що в браузері *driver1*, присутні три коментаря — *comment1, comment2* та *comment3*.
   4. В браузері *driver2* видалити коментар *comment2.* Протестувати, що в браузері *driver2* присутні два коментаря — *comment1* та *comment3*.
   5. Закрити обидва браузера, після чого знову запустити браузер і протестувати, що анонімний користувач бачить два коментаря — *comment1* та *comment3*.

Тестування необхідно організувати за допомогою фреймворку unittest та з використанням Selenium WebDriver.

Схема взаємодії тестового модуля з клієнтською та серверною частинами веб-застосунку показана на рисунку. В загальному випадку веб-застосунок використовує базу даних, але для спрощення в завданні не передбачається використання БД.

# Специфікація веб-застосунку

Веб-застосунок призначений для отримання коментарів-відповідей на запитання, складається з однієї веб-сторінки і має наступну функціональність:

1. Довільний користувач може переглянути питання, яке розташоване в лівій верхній частині сторінки, та існуючі коментарі до цього питання.
2. ***Sign Up:*** новий користувач може зареєструватися, увівши свої значення в поля *Display Name, Email, Password* та натиснувши кнопку **Sign Up.** При виконання умови, що всі поля заповнені і значення *Email –* унікальне серед всіх e-mail на *backend*, реєстрація буде виконана успішно, користувач автоматично входить в свій обліковий запис, і на сторінці з’являється кнопка **Log Out.**
3. Користувач може вийти зі свого облікового запису, натиснувши кнопку **Log Out.**
4. ***Log In:*** зареєстрований користувач може увійти в свій обліковий запис. Для цього необхідно ввести правильні e-mail та пароль в поля *Email* та *Password* , після чого натиснути кнопку **Log In.**
5. Користувач, що увійшов в свій обліковий запис, може додавати коментарі до питання. Для цього користувачу необхідно написати текст коментаря в спеціальному редакторі, що знаходиться на сторінці, та натиснути кнопку **New comment.**
6. До натискання кнопки **New comment** користувач може форматувати текст коментаря за допомогою кнопок редактора – виділити коментар або його частину **жирним** (кнопка **B**), *курсивом* (кнопка *I*), підкресленим (кнопка *U*) або ~~закресленим~~ (кнопка ~~O~~).
7. Користувач, що увійшов в свій обліковий запис, може видаляти створений ним коментар, натиснувши кнопку **Remove,** розташовану справа від потрібного коментаря.
8. За замовчуванням на сторінці присутні коментарі “Test comment 1”, “Test comment 2”, створені користувачами Alice A. та Bob B.,які мають наступні *Email* та *Password*:

[alice\_2002@gmail.com](mailto:alice_2002@gmail.com), `*aaa*`

[bob\_2001@gmail.com](mailto:bob_2001@gmail.com), `*bbb*`.

1. Для спрощення задачі веб-застосунок не зберігає дані постійно. При зупинці *backend* всі дані будуть втрачені. Тестові користувачі та коментарі створюються автоматично при кожному запуску *backend*.

**Для запуску веб-застосунку необхідно:**

1. Перед першим запуском веб-застосунку встановити бібліотеку Tornado – спеціальний веб-фреймворк Python (асинхронний сервер) за допомогою команди

pip install tornado .

1. Запустити *backend.py* .
2. Звернутися до веб-застосунку за URL [*http://localhost:8000*](http://localhost:8000) *.*

# Висновки

В ході виконання курсової роботи було розроблено програму тестування локального веб-застосунку з використанням Python та Selenium WebDriver, бібліотеки unittest.

В результаті тестування виявлено, що розроблені програми повністю відповідають специфікаціям та працюють коректно.

# Список літератури

1. Майерс Г. Искусство тестирования программ / Пер. с англ. под ред. Б. А. Позина – М.: Финансы и статистика, 1982. – 176 с.
2. Дідковська М.В. Технологія розробки та тестування програм: Навчальний посібник / М.В. Дідковська. – К.: «Вік-принт», 2014. – 344 с.
3. Lewis W. E. Software testing and continuous quality improvement / W. E. Lewis – 3rd ed. – CRC Press, 2009. – 640 p.
4. Patton R. Software testing / R. Patton. – Indianapolis: Sams, 2005. – 408 p.

# Додаток 2. Текст програми тестування

from selenium import webdriver

from selenium.webdriver.common.by import By

from subprocess import Popen

import time

import unittest

class MainTestClass(unittest.TestCase):

BACKEND\_NAME = 'backend.py'

START\_URL = 'http://localhost:8001'

LOGOUT\_SIGNATURE = 'Log Out'

def setUp(self):

self.backend\_process = Popen(['python', self.BACKEND\_NAME])

def tearDown(self):

self.backend\_process.kill()

def \_log\_in(self, driver, email, password):

elem = driver.find\_element(By.NAME, 'email')

elem.clear()

elem.send\_keys(email)

elem = driver.find\_element(By.NAME, 'password')

elem.clear()

elem.send\_keys(password)

driver.find\_element(By.XPATH, '//button[.="Log In"]').click()

def \_post\_comment(self, driver, text\_comment):

driver.find\_element(By.CLASS\_NAME, 'ql-editor').clear()

driver.find\_element(By.CLASS\_NAME, 'ql-editor').send\_keys(text\_comment)

driver.find\_element(By.XPATH, '//button[.="New comment"]').click()

def test\_task1(self):

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

elem = driver.find\_elements(By.XPATH, '//h1[.="Is this a good way to process input?"]')

self.assertTrue(elem, 'Question header is absent')

elem = driver.find\_elements(By.XPATH, '//div[@id="question"]/pre[text()!=""]')

self.assertTrue(elem, 'Question body is empty')

comments = (

('Test comment 1', 'Alice A.'),

('Test comment 2', 'Bob B.'),

)

for comment, elem in zip(comments, driver.find\_elements(By.XPATH, '//div[@id="comments"]/ul/li')):

comment\_text = elem.find\_element(By.TAG\_NAME, 'span').text

author = elem.find\_element(By.TAG\_NAME, 'a').text

self.assertEqual(comment, (comment\_text, author), 'Comment does not match')

elem = driver.find\_elements(By.XPATH, '//button[.="Sign Up"]')

self.assertTrue(elem, 'Sign up button is missing')

elem = driver.find\_elements(By.XPATH, '//button[.="Log In"]')

self.assertTrue(elem, 'Log in button is missing')

elem = driver.find\_elements(By.XPATH, '//div[input[@name="display\_name"]]')

self.assertTrue(elem, 'There is no field for entering a Display name')

elem = driver.find\_elements(By.XPATH, '//div[input[@name="email"]]')

self.assertTrue(elem, 'There is no field for entering an Email')

elem = driver.find\_elements(By.XPATH, '//div[input[@name="password"]]')

self.assertTrue(elem, 'There is no field for entering a Password')

time.sleep(5)

def test\_task2(self):

display\_name = 'Cavin C.'

email = 'cavin@gmail.com'

password = 'ccc'

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

driver.find\_element(By.NAME, 'display\_name').send\_keys(display\_name)

driver.find\_element(By.NAME, 'email').send\_keys(email)

driver.find\_element(By.NAME, 'password').send\_keys(password)

driver.find\_element(By.XPATH, '//button[.="Sign Up"]').click()

raw\_text = driver.find\_element(By.ID, 'signup-section').text

self.assertEqual(raw\_text[:-len(self.LOGOUT\_SIGNATURE)], display\_name)

driver.find\_element(By.XPATH, '//button[.="Log Out"]').click()

elems = driver.find\_elements(By.XPATH, '//button[.="Log In"]')

self.assertTrue(elems, 'User did not log out')

other\_display\_name = 'Bella B.'

elem = driver.find\_element(By.NAME, 'display\_name')

elem.clear()

elem.send\_keys(other\_display\_name)

elem = driver.find\_element(By.NAME, 'email')

elem.clear()

elem.send\_keys(email)

elem = driver.find\_element(By.NAME, 'password')

elem.clear()

elem.send\_keys(password)

driver.find\_element(By.XPATH, '//button[.="Sign Up"]').click()

elems = driver.find\_elements(By.XPATH, '//button[.="Log Out"]')

self.assertFalse(elems, 'User actually logged in with non unique email')

time.sleep(5)

def test\_task3(self):

email, password = 'alice\_2002@gmail.com', 'aaa'

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

self.\_log\_in(driver, email, password)

self.assertTrue(driver.find\_elements(By.XPATH, '//button[.="Log Out"]'))

raw\_text = driver.find\_element(By.ID, 'signup-section').text

self.assertEqual(raw\_text[:-len(self.LOGOUT\_SIGNATURE)], 'Alice A.')

def test\_task4(self):

email, password = 'alice\_2002@gmail.com', 'aaa'

my\_script = """

var elem = document.getElementById('editor-section');

elem.style.borderColor = "red";

elem.style.borderStyle = "solid";

"""

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

self.\_log\_in(driver, email, password)

elem = driver.find\_element(By.ID, 'editor-section')

self.assertTrue(elem, 'Editor section is not present')

elem = driver.find\_element(By.XPATH, '//button[.="New comment"]')

self.assertTrue(elem, 'New comment button is missing')

driver.execute\_script(my\_script)

driver.save\_screenshot('my\_screeen.png')

def test\_task5(self):

def click\_style(style\_name):

driver.find\_element(By.CLASS\_NAME, 'ql-' + style\_name).click()

email, password = 'alice\_2002@gmail.com', 'aaa'

my\_comment = 'Comment from Alice'

expected\_formatted\_comment\_html = '<span><u>Comment</u> from Alice</span>'

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

self.\_log\_in(driver, email, password)

self.\_post\_comment(driver, my\_comment)

xp = '//div[@id="comments"]/ul/li[last()][span="{}"][a="Alice A."][span/button="Remove"]'.format(my\_comment)

self.assertTrue(driver.find\_element(By.XPATH, xp), 'New comment is not correct')

driver.find\_element(By.CLASS\_NAME, 'ql-editor').clear()

elem = driver.find\_element(By.CLASS\_NAME, 'ql-editor')

click\_style('underline')

elem.send\_keys('Comment')

click\_style('underline')

elem.send\_keys(' from Alice')

driver.find\_element(By.XPATH, '//button[.="New comment"]').click()

elem = driver.find\_element(By.XPATH, '//div[@id="comments"]/ul/li[last()]/span')

extracted\_html = elem.get\_attribute('outerHTML')

self.assertEqual(extracted\_html, expected\_formatted\_comment\_html)

#italic

expected\_formatted\_comment\_html = '<span><em>Comment</em> from Alice</span>'

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

self.\_log\_in(driver, email, password)

self.\_post\_comment(driver, my\_comment)

xp = '//div[@id="comments"]/ul/li[last()][span="{}"][a="Alice A."][span/button="Remove"]'.format(my\_comment)

self.assertTrue(driver.find\_element(By.XPATH, xp), 'New comment is not correct')

driver.find\_element(By.CLASS\_NAME, 'ql-editor').clear()

elem = driver.find\_element(By.CLASS\_NAME, 'ql-editor')

click\_style('italic')

elem.send\_keys('Comment')

click\_style('italic')

elem.send\_keys(' from Alice')

driver.find\_element(By.XPATH, '//button[.="New comment"]').click()

elem = driver.find\_element(By.XPATH, '//div[@id="comments"]/ul/li[last()]/span')

extracted\_html = elem.get\_attribute('outerHTML')

self.assertEqual(extracted\_html, expected\_formatted\_comment\_html)

def test\_task6(self):

email, password = 'alice\_2002@gmail.com', 'aaa'

my\_comment1 = 'aaaaaaaaaaaaaaaaaaa'

my\_comment2 = 'AAAAAAAAAAAAAAAAAAA'

driver = webdriver.Chrome()

with driver:

driver.get(self.START\_URL)

self.\_log\_in(driver, email, password)

self.\_post\_comment(driver, my\_comment1)

xp = '//div[@id="comments"]/ul/li[last()][span="{}"][a="Alice A."][span/button="Remove"]'.format(

my\_comment1)

self.assertTrue(driver.find\_element(By.XPATH, xp), 'New comment is not correct')

self.\_post\_comment(driver, my\_comment2)

xp = '//div[@id="comments"]/ul/li[last()][span="{}"][a="Alice A."][span/button="Remove"]'.format(

my\_comment2)

self.assertTrue(driver.find\_element(By.XPATH, xp), 'New comment is not correct')

def test\_task7(self):

display\_name,existing\_email, existing\_password = 'Alice A.', 'alice\_2002@gmail.com', 'aaa'

driver = webdriver.Chrome()

with driver:

#a. Переконатися, що видаляється потрібний коментар.

# b. Переконатися, що усі інші коментарі, створені даним користувачем та іншими користувачами, залишилися на місці.

driver.get(self.START\_URL)

self.\_log\_in(driver, existing\_email, existing\_password)

count\_comments\_old = driver.find\_elements\_by\_xpath(f'//div[@id="comments"]/ul/li')

all\_comments = driver.find\_elements\_by\_xpath(

f'//div[@id="comments"]/ul/li[a = "{display\_name}"]/span[1]')

users\_comments\_text = [comment.text for comment in all\_comments]

del\_comm = users\_comments\_text[0]

remove\_1\_comm\_button = driver.find\_elements\_by\_xpath(f'//div[@id="comments"]/ul/li[span="{del\_comm}"][a="{display\_name}"]/span/button[.="Remove"]')[0]

remove\_1\_comm\_button.click()

count\_comments\_new = driver.find\_elements\_by\_xpath(f'//div[@id="comments"]/ul/li')

self.assertEqual(len(count\_comments\_new), len(count\_comments\_old) - 1, 'Problems in deleting comment')

def test\_task8(self):

driver = webdriver.Chrome('/usr/local/bin/chromedriver')

another\_driver = webdriver.Chrome('/usr/local/bin/chromedriver')

with driver as driver1:

driver1.get(self.START\_URL)

inpt1 = driver1.find\_element\_by\_xpath('//input[@name="display\_name"]')

inpt1.send\_keys("user1")

inpt2 = driver1.find\_element\_by\_xpath('//input[@name="email"]')

inpt2.send\_keys("user1@gmail.com")

inpt3 = driver1.find\_element\_by\_xpath('//input[@name="password"]')

inpt3.send\_keys("111")

time.sleep(1)

driver1.find\_element\_by\_xpath('//button[.="Sign Up"]').click()

time.sleep(1)

field = driver1.find\_element\_by\_xpath('//div[@class="ql-container ql-snow"]/div/p[.="(enter new comment)"]')

field.clear()

comment1\_txt = "comment1"

driver1.find\_element\_by\_xpath('//div[@class="ql-container ql-snow"]/div').send\_keys(comment1\_txt)

driver1.find\_element\_by\_xpath('//button[.="New comment"]').click()

time.sleep(2)

with another\_driver as driver\_test:

driver\_test.get(self.START\_URL)

inpt1 = driver\_test.find\_element\_by\_xpath('//input[@name="display\_name"]')

inpt1.clear()

inpt1.send\_keys("user2")

inpt2 = driver\_test.find\_element\_by\_xpath('//input[@name="email"]')

inpt2.clear()

inpt2.send\_keys("user2@gmail.com")

inpt3 = driver\_test.find\_element\_by\_xpath('//input[@name="password"]')

inpt3.clear()

inpt3.send\_keys("222")

time.sleep(1)

driver\_test.find\_element\_by\_xpath('//button[.="Sign Up"]').click()

time.sleep(1)

field = driver\_test.find\_element\_by\_xpath(

'//div[@class="ql-container ql-snow"]/div/p[.="(enter new comment)"]')

field.clear()

comment2\_txt = "comment2"

driver\_test.find\_element\_by\_xpath('//div[@class="ql-container ql-snow"]/div').send\_keys(comment2\_txt)

driver\_test.find\_element\_by\_xpath('//button[.="New comment"]').click()

time.sleep(3)

lst\_comments = list()

lst\_comments.append(comment1\_txt)

lst\_comments.append(comment2\_txt)

index\_lst = [-2, -1]

path\_page\_comments = driver\_test.find\_elements\_by\_xpath('//div[@id="comments"]/ul/li/span[1]')

lst\_page\_comments = [path\_page\_comments[i].get\_attribute("innerHTML") for i in index\_lst]

self.assertListEqual(lst\_page\_comments, lst\_comments, "Comment structure was broken")

time.sleep(5)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

# Додаток 6. Скріншоти результатів виконання тестів

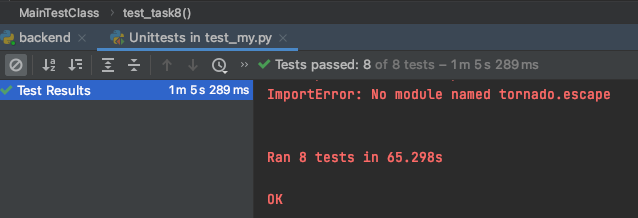


Рис. 1 – Результат виконання тестів