НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №4
із дисципліни «Автоматизоване тестування програмного забезпечення»
на тему
Тестування веб-застосунку за допомогою Python та Selenium WebDriver

Виконав:
студент групи КМ-82
Бубела Д. В.

Керівник:
асистент
Громова В. В.

Київ — 2020

ЗМІСТ

МЕТА РОБОТИ

Ознайомитися з «Selenium WebDriver». Протестувати локально веб-застосунок простої структури відповідно до специфікації. Тестування здійснювати в python за допомогою модуля unittest з застосуванням «Selenium WebDriver»

Специфікація

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, ‘aaa‘ bob_2001@gmail.com, ‘bbb‘.

9. 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.

1 ПОСТАНОВКА ЗАДАЧІ

Необхідні для встановлення пакети в пайтон: tornado, selenium. Selenium WebDriver відповідно до браузера встановлюється в систему.

Завдання

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

а) 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.

б) 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.

в) 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.

г) 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.

д) 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.

е) 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.

ж) 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.

з) 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.

2 ОСНОВНА ЧАСТИНА

В обох завданнях уся робота здійснюється в python з бібліотекою для тестуваннь unittest. Лабораторна робота проводиться в браузері «Mozilla Firefox 83.0». Для здійснення приймального тестування використовувався модуль Popen з бібліотеки subprocess. Він потрібен для створення нового процесу від запущеного скрипта, що забезпечував би роботу веб сервера, який тестується.

Метод setUp створює процес “бекенду” і ініціалізує веб-драйвер. Метод tearDown завершує роботу веб-драйвера та сервера, що тестується. Ці методи викликаються безпосередньо перед та після виконання кожного з тестів.

Під час тестування в консоль виводяться помилки сервера, але так має бути. Unittest перенаправляє всі вихідні потоки підпроцесу у вивід тої програми, яка запущеня для тестування.

Для коректного збереження скріншоту в завданні г) необхідно додати таймаут time.sleep(2).

ВИСНОВКИ

Ми навчилися використовувати Selenium WebDriver для приймального тестування веб-додатків. У ході роботи було розроблено 8 тест-кейсів для повного покриття специфікації програми. Також попрактикувались в створенні автоматичних тестів з допомогою unittest та виклику окремих інструкцій до та після виконання окремих тестів. Переглянули способи використання конструкцій __enter__ та __exit__ що було необхідно для розуміння роботи викликів вебдрайверу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Python Software Foundation. subprocess — Subprocess management [Електронний ресурс] / Python Software Foundation.. — 2020. — Режим доступу до ресурсу: https://docs.python.org/3/library/subprocess.html..

2. Selenium IDE API Reference [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.selenium.dev/selenium-ide/docs/en/api/commands.

3. MDN contributors. XSLT/XPath Reference [Електронний ресурс] / MDN contributors. – 2020. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/XPath.

4. Python Software Foundation.unittest — Unit testing framework[Електронний ресурс] / Python Software Foundation.. — 2020. — Режим доступу до ресурсу: https://docs.python.org/3/library/unittest.html.

5. FriendFeed. Structure of a Tornado web application [Електронний ресурс] / FriendFeed // 6.1.0. – 2019. – Режим доступу до ресурсу: https://www.tornadoweb.org/en/stable/guide/structure.html.

Додаток А

Текст модулів, що реалізують автоматичне тестування

Лістинг файлу test.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from subprocess import Popen
import time
import unittest


class AppTest(unittest.TestCase):
    BACKEND_NAME = 'backend.py'
    START_URL = 'http://localhost:8000'
    LOGOUT_SIGNATURE = 'Log Out'

    def setUp(self):
        self.backend_process = Popen(['python', self.BACKEND_NAME])
        self.driver = webdriver.Firefox()

    def tearDown(self):
        # Despite of it's already in __exit__, memory won't free idk
        self.driver.quit()
        self.backend_process.kill()
        self.backend_process.wait()

    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 _create_user(self, driver, name, email, password):
        name_field = driver.find_element(By.NAME, 'display_name')
        email_field = driver.find_element(By.NAME, 'email')
        password_field = driver.find_element(By.NAME, 'password')

        # unary clearing doesn't work in multiple sessions
        name_field.send_keys()
        name_field.clear()
        name_field.send_keys(name)
        email_field.send_keys()
        email_field.clear()
        email_field.send_keys(email)
        password_field.send_keys()
        password_field.clear()
        password_field.send_keys(password)

        driver.find_element(By.XPATH, '//button[.="Sign Up"]').click()
        btn = driver.find_element(By.XPATH, '//button[.="Log Out"]')
        btn.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 _check_last_comment(self, driver, comment, author):
        xp = f'//div[@id="comments"]/ul/li[last()][span="{comment}"]'\
            f'[a="{author}"][span/button="Remove"]'
        self.assertTrue(
            driver.find_element(By.XPATH, xp),
            'New comment is not correct')

    def test_1_anonymous(self):
        """
        1. Перевіряє, що анонімний користувач бачить на сторінці
        """
        with self.driver as driver:
            driver.get(self.START_URL)
            # a
            elems = driver.find_elements(
                By.XPATH, '//h1[.="Is this a good way to process input?"]')
            self.assertTrue(elems, 'Question header is absent')

            # b
            elems = driver.find_elements(
                By.XPATH, '//div[@id="question"]/pre[text()!=""]')
            self.assertTrue(elems, 'Question body is empty')

            # c
            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')

        # d
        elems = driver.find_elements(By.XPATH, '//button[.="Sign Up"]')
        self.assertTrue(elems, 'Sign up button is missing')

        elems = driver.find_elements(By.XPATH, '//button[.="Log In"]')
        self.assertTrue(elems, 'Log In button is missing')

        # e
        # Display Name
        elems = driver.find_elements(
            By.XPATH, '//div[input[@name="display_name"]]')
        self.assertTrue(
            elems, 'There is no field for entering a Display name')
        elems = driver.find_elements(
            By.XPATH, '//div[label[@for="name"]/b]')
        self.assertTrue(
            elems, 'There is no label for a Display name')
        # Email
        elems = driver.find_elements(
            By.XPATH, '//div[input[@name="email"]]')
        self.assertTrue(
            elems, 'There is no field for entering an Email')
        elems = driver.find_elements(
            By.XPATH, '//div[label[@for="email"]/b]')
        self.assertTrue(
            elems, 'There is no label for an Email')
        # Password
        elems = driver.find_elements(
            By.XPATH, '//div[input[@name="password"]]')
        self.assertTrue(
            elems, 'There is no field for entering a Password')
        elems = driver.find_elements(
            By.XPATH, '//div[label[@for="psw"]/b]')
        self.assertTrue(
            elems, 'There is no label for a Password')

def test_2_signup(self):
    """
    2. Перевіряє, що новий користувач може зареєструватись
    """
    display_name = 'Carol C.'
    email = 'carol@gmail.com'
    password = 'ccc'

    with self.driver as driver:
        driver.get(self.START_URL)
        # a
        # signing up
        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()
        # Name check
        raw_text = driver.find_element(By.ID, 'signup-section').text
        self.assertEqual(
            raw_text[:-len(self.LOGOUT_SIGNATURE)], display_name)
        # Log out usage check
        btn = driver.find_element(By.XPATH, '//button[.="Log Out"]')
        # b
        btn.click()
        elems = driver.find_elements(By.XPATH, '//button[.="Log In"]')
        self.assertTrue(elems, 'User did not log out')
        # c
        other_display_name = 'Borya 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')

def test_3_log_in(self):
    """
    3. Перевіряє, що зареєстрований користувач входить до свого облікового запису
    """
    email, password = 'alice_2002@gmail.com', 'aaa'
```

```
    with self.driver as 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.')
        driver.find_element(By.XPATH, '//button[.="Log Out"]').click()

        self._log_in(driver, email, 'wrong-password')
        elems = driver.find_elements(By.XPATH, '//button[.="Log Out"]')
        self.assertFalse(
            elems, 'User actually logged in with wrong password')

        self._log_in(driver, 'wrong-email', password)
        elems = driver.find_elements(By.XPATH, '//button[.="Log Out"]')
        self.assertFalse(
            elems, 'User actually logged in with wrong email')

def test_4_editor(self):
    """
    4. Перевіряє, що на сторінці є Редактор та кнопка New comment
    """
    email, password = 'alice_2002@gmail.com', 'aaa'
    my_script = """
    var elem = document.getElementById('editor-section');
    elem.style.borderColor = "red";
    elem.style.borderStyle = "solid";
    """
    with self.driver as 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('comments_screenshot.png')
        time.sleep(2)

def test_5_comment(self):
    """
    5. Перевіряє, що користувач, може ввести новий коментар
    """
    def click_style(style_name):
        driver.find_element(By.CLASS_NAME, 'ql-' + style_name).click()

    name = "Alice A."
    email, password = 'alice_2002@gmail.com', 'aaa'
    my_comment = 'Comment from Alice'
    expected_formatted_comment_html = '<span><strong>Comment</strong> '\
        '<em>from <s>Alice</s></em> <u>Dmytro</u></span>'

    with self.driver as driver:
        driver.get(self.START_URL)
        self._log_in(driver, email, password)
        # a
        self._post_comment(driver, my_comment)
        self._check_last_comment(driver, my_comment, name)
        # b
        text_field = driver.find_element(By.CLASS_NAME, 'ql-editor')
        text_field.clear()
        click_style('bold')
        text_field.send_keys('Comment')
        click_style('bold')
        text_field.send_keys(' ')
        click_style('italic')
        text_field.send_keys('from ')
        click_style('strike')
        text_field.send_keys('Alice')
        click_style('strike')
        click_style('italic')
        text_field.send_keys(' ')
        click_style('underline')
        text_field.send_keys('Dmytro')
        click_style('underline')
        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)
        # c
        self._check_last_comment(driver, "Comment from Alice Dmytro", name)

def test_6_comment(self):
    """
    6. Перевіряє, що користувач, може ввести ще один коментар
    """
    name = "Alice A."
    email, password = 'alice_2002@gmail.com', 'aaa'
    comments = ('Previous Alice comment', 'Current Alice comment')
    with self.driver as driver:
        driver.get(self.START_URL)
        self._log_in(driver, email, password)
        # a
        for comment in comments:
```

```
            self._post_comment(driver, comment)
            self._check_last_comment(driver, comment, name)
        # b
        for comment, elem in zip(
            comments,
            driver.find_elements(By.XPATH, '//div[@id="comments"]/ul/li[position() > (last() - 2)]')
        ):
            comment_text = elem.find_element(By.TAG_NAME, 'span').text
            author = elem.find_element(By.TAG_NAME, 'a').text
            self.assertEqual((comment, name), (comment_text, author),
                             'Comment does not match')

def test_7_remove(self):
    """
    7. Перевіряє, що користувач може видалити створений ним коментар
    """
    email, password = 'alice_2002@gmail.com', 'aaa'
    comment = 'Alice comment'
    with self.driver as driver:
        driver.get(self.START_URL)
        self._log_in(driver, email, password)
        comments_before = driver.find_elements(
            By.XPATH,
            '//div[@id="comments"]/ul/li')
        self._post_comment(driver, comment)
        driver.find_element(By.XPATH, '//div[@id="comments"]/ul/li[last()]/span/button[.="Remove"]').click()
        comments_after = driver.find_elements(
            By.XPATH,
            '//div[@id="comments"]/ul/li')
        self.assertEqual(comments_before, comments_after)

def test_8_multiplayer(self):
    """
    Перевіряє, що можуть бути активними два різних користувача
    """
    user_1 = ('User 1', 'u1@gmail.com', 'userone')
    user_2 = ('User 2', 'u2@gmail.com', 'usertwo')
    # create user 1, 2
    with webdriver.Firefox() as driver:
        driver.get(self.START_URL)
        self._create_user(driver, *user_1)
        self._create_user(driver, *user_2)

    with webdriver.Firefox() as driver_1, webdriver.Firefox() as driver_2:
        driver_1.get(self.START_URL)
        driver_2.get(self.START_URL)
        # a
        self._log_in(driver_1, *user_1[1:])
        self._post_comment(driver_1, 'comment1')
        # b
        self._log_in(driver_2, *user_2[1:])
        self._post_comment(driver_2, 'comment2')

        comment_2 = driver_2.find_elements(
            By.XPATH,
            '//div[@id="comments"]/ul/li/span[.="comment2"]')
        self.assertTrue(comment_2, 'Comment 2 is absent')
        comment_1 = driver_2.find_elements(
            By.XPATH,
            '//div[@id="comments"]/ul/li/span[.="comment1"]')
        self.assertTrue(comment_1, 'Comment 1 is absent')
        # c
        self._post_comment(driver_1, 'comment3')
        self.assertTrue(
            driver_1.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[.="comment1"]'),
            'Comment 1 is absent')
        self.assertTrue(
            driver_1.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[.="comment2"]'),
            'Comment 2 is absent')
        self.assertTrue(
            driver_1.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[.="comment3"]'),
            'Comment 3 is absent')
        # d
        driver_2.find_element(
            By.XPATH,
            '//div[@id="comments"]/ul/li[span = "comment2"]/span/button[.="Remove"]'
        ).click()
        self.assertTrue(
            driver_2.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[.="comment1"]'),
            'Comment 1 is absent')
        self.assertTrue(
            driver_2.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[.="comment3"]'),
            'Comment 3 is absent')
    # e
    with webdriver.Firefox() as driver:
        driver.get(self.START_URL)
        self.assertTrue(
            driver.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li[span="comment1"]'),
            'Comment 1 is absent')
```

```
            self.assertTrue(
                driver.find_elements(
                    By.XPATH,
                    '//div[@id="comments"]/ul/li[span="comment3"]'),
                'Comment 3 is absent')


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

Додаток Б

Скріншоти результатів виконання автотестів

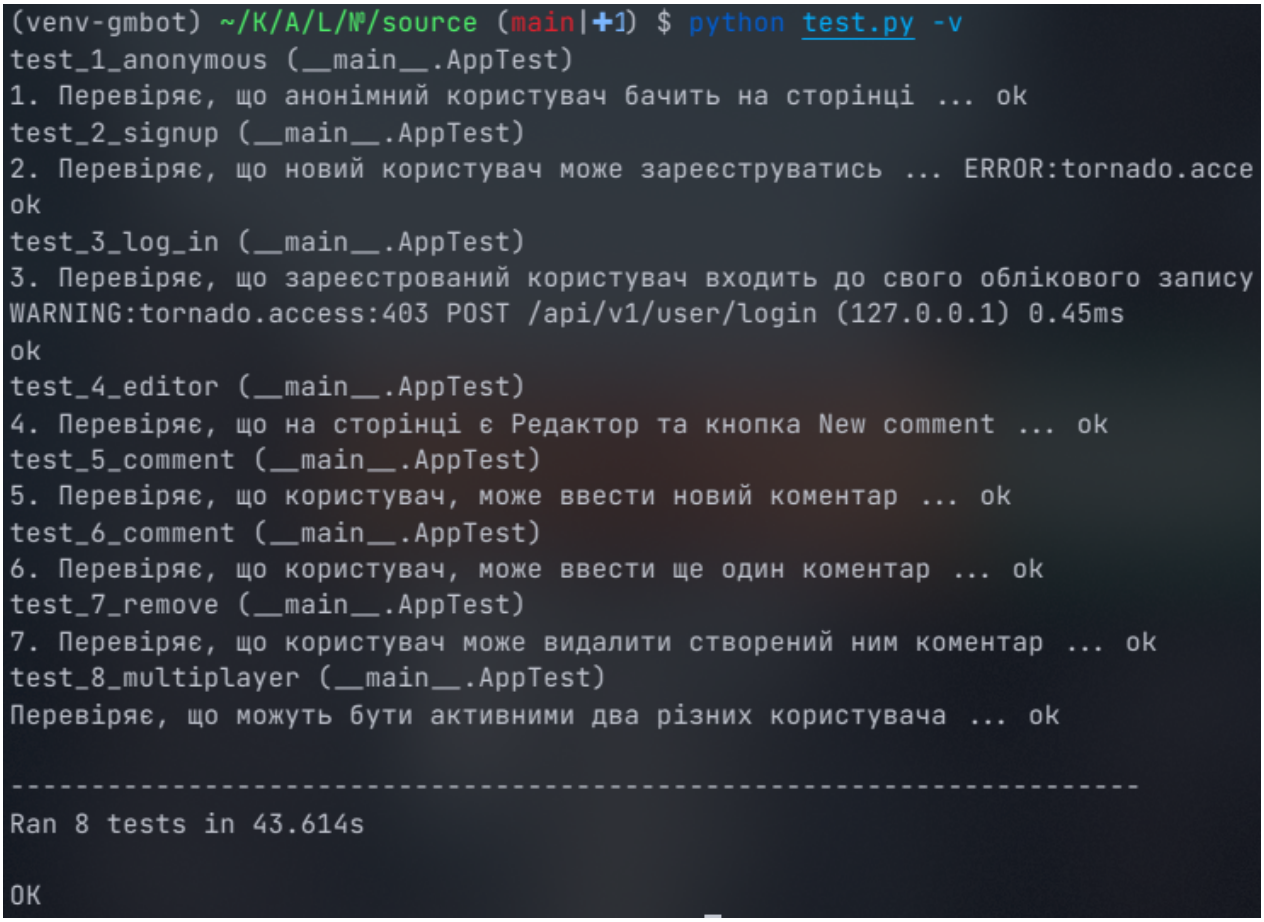

Рисунок Б.1 – Скріншот тестування test_1.py

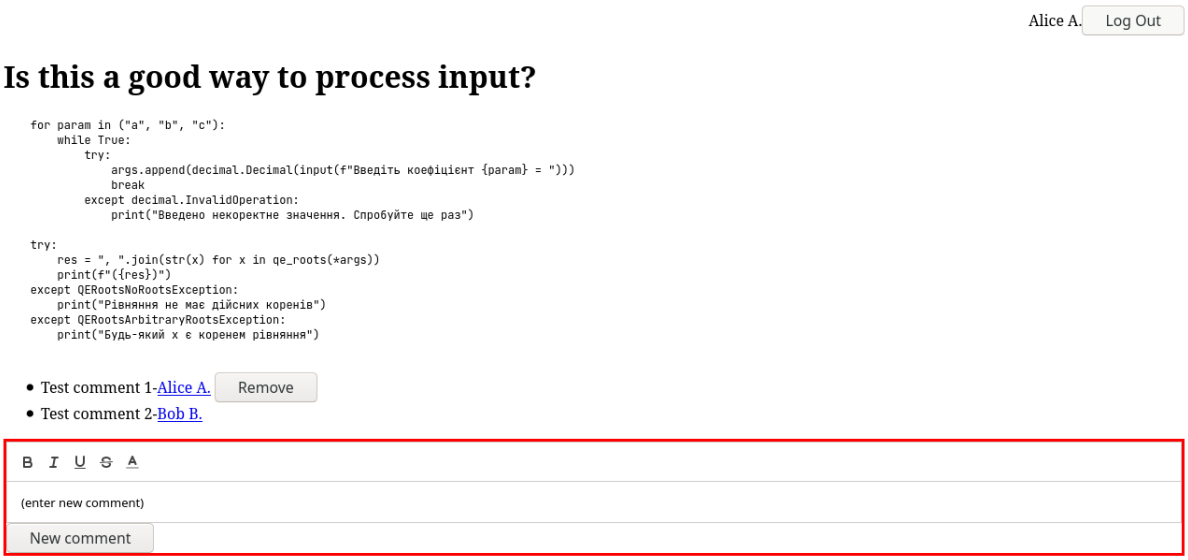

Рисунок Б.2 – Скріншот зроблений за допомогою Selenium WebDriver comments_screenshot.png

Додаток В

Відповіді на контрольні запитання

1) Що таке Selenium WebDriver? Яке його призначення?

Webdriver — це інструмент управління браузером за допомогою його API, з системи проддуктів Selenium, що використовується в зв’язці з мовами програмування.

2) Як запустити браузер за допомогою Python та Selenium WebDriver?

Потрібно ініціалізувати вебдрайвер та за його допомогою

3) Як ввести текст в текстове поле за допомогою Selenium WebDriver?

Знайти через драйвер і відповідні його методи текстове поле, додати туди текст методом send_keys або іншим доступним, влючно з наведенням курсору і імітування натискання клавіш клавіатури, викликом JavaScript.

4) Як за допомогою Python та Selenium WebDriver перевірити, що певний елемент відображається на екрані?

Спробувати його знайти на сторінці (метод find_element) і перевірити чи пошук виявився успішним.

5) Як можна зачекати появу елемента на сторінці?

Можна використати метод WebDriverWait з selenium.webdriver.support.ui та expected_conditions з elenium.webdriver.support наприклад element = WebDriverWait(driver, 10) .until(EC.presence_of_element_located((By.ID, "myDynamicElement"))) або просто зачекати методами python наприклад time.sleep(10).

6) Як можна отримати текст веб-елемента? Як можна отримати атрибут веб-елемента?

Текста можна отримати за допомогою метода елемента (WebElement) — text.

Атрибут можна отримати за допомогою метода елемента — get_attribute("attribute name")

7) Коли використовуються findElement() та findElements()?

Перший метод повертає елемент першого співпадіння з критерієм умови. Другий метод повертає всі елементи, що задовольняють критерій у вигляді списку з WebElement.

8) Як можна зробити скріншот за допомогою Python та Selenium WebDriver?

За допомогою методу вебдрайвера — save_screenshot.

9) Як і з якою метою можна виконати JavaScript код в браузері за допомогою Selenium WebDriver?

Можна використати метод драйвера execute_script аргументом якого передати необхідний для виконання скрипт.

10) Наведіть переваги та недоліки Selenium WebDriver, які Ви помітили під час виконання лабораторної роботи?

Повна імітація роботи користувача за допомогою вбудованого API браузера має як переваги так і недоліки. Можливість використовувати її за допомогою мови програмування є великою перевагою, оскільки разом з викликом js забезпечує велику гнучкість в роботі. До недоліків можна віднести: Нестабільність роботи, особливо різна поведінка на різних браузерах з різними драйверами та різними рушіями виконання JavaScript, швидкість виконання.