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

Курсова робота
із дисципліни «Автоматизоване тестування програмного забезпечення»

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

Керівник:
старший викладач
Бай Ю. П.

Київ — 2020

ЗМІСТ

МЕТА РОБОТИ

Узагальнити знання отримані з курсу «Автоматизоване тестування програмного забезпечення». Протестувати веб-застосунок на модульному рівні за допомогою бібліотеки unittest, на приймальному рівні за допомогою Selenium WebDriver та http запитів не використовуючи Selenium, за допомогою бібліотеки requests.

Розробити специфікацію до веб-застосунку. Перевірити роботу тестових модулів.

1 РЕАЛІЗАЦІЯ ЗАВДАННЯ

1.1 Умова завдання

Завдання 2

4) В одновимірному масиві, що складається з n елементів, обчислити середньоквадратичне значення елементів масиву.

Програма приймає на вхід список чисел. Якщо деякі з них не існують, значення None, вони пропускаються. На пустий масив відповідає нічим (None).

1.2 Специфікація веб-застосунку

Специфікація

Веб-застосунок призначений для отримання відповідей на введені вхідні дані відповідно до завдання описаного вище, складається з однієї веб-сторінки і має наступну функціональність:

1. Довільний користувач може переглянути умову завдання, яке розташоване в центральній верхній частині сторінки, та існуючі відповіді до цього завдання.

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 вводити вхідні дані для завдання та отримувати відповідь. Для цього користувачу необхідно написати в полях вхідні числові дані, та натиснути кнопку **Обчислити**.

6. До натискання кнопки **Обчислити** користувач може додавати нові числа у спливаючі поля та видаляти дані в існуючих.

7. Користувач, що увійшов в свій обліковий запис, може видаляти створену ним відповідь, натиснувши кнопку **Remove**, розташовану справа від потрібної відповіді.

8. За замовчуванням на сторінці присутні дані результатів попередніх виконань завдань “Input: [2, 4, 16, 256] Answer: 128.2692480682724”, “Input: [1, -1, 4, -4] Answer: 2.915475947422650 4”, створені користувачами Linus T. та Iryna K., які мають наступні Email та Password: torvalds@osdl.org, ‘kernel‘ kostushkoia5@gmail.com, ‘numericalerror‘.

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.

1.3 Розробка і реалізація тест-кейсів

1.3.1 test_1.py

Таблиця 1.1 – Тест-кейси test_1

Тест	Мета	Вхідні данні	Очікуваний результат
test_valid_input	Перевірити виконання на цілих числах	[3, 13, 33, 87]	47
test_zeros	Перевірити виконання з нулями	[0, 0, 0]	0
test_negative	Перевірити виконання з від'ємними числами	[-99, -99, -1]	80.83522334
test_real	Перевірити виконання функції з дійсними числами	[0.3331, 5.032, -8989]	5189.80238647
test_no_input	Перевірити виконання функції без даних	[]	None
test_type_error	Перевірити виконання функції з некоректними даними	['1', 90, 4]	TypeError

1.3.2 test_2.py

Тестування за допомогою http запитів. Бібліотека requests

а) Тестування входу зареєстрованого користувача.

1) Чи присутнє ім'я у відповіді на вхід.

2) Чи присутній JSON Web Token у відповіді. Він необхідний для здійснення авторизованих запитів.

3) Чи присутнє id користувача у відповіді

б) Перевірка входу з невірним паролем

в) Перевірка входу з невірною поштою

г) Чи можна створити допис з відповіддю від користувача, даних для входу якого ми не знаємо.

д) Чи можна вийти з облікового запису. Перевірка відповіді від кнопки **Log Out**

е) Перевірка створення коментаря з аккаунту зареєстрованого користувача. Перевірка всіх дописів залишених на сайті. Перевірка відповіді.

ж) Перевірка реєстрації нового користувача.

з) Чи може користувач зареєструватися повторно з однаковими даними.

1.3.3 test_3.py

Тестування за допомогою Selenium WebDriver

а) Перевіряє, що анонімний користувач бачить на сторінці:

1) Напис "Курсова робота".

2) Заголовок питання, який співпадає з очікуваним заголовком.

3) Попередні відповіді, 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 поля для введення вхідних даниз на завдання та кнопка **Обчислити**.

д) Перевіряє, що користувач, який знаходиться у своєму обліковому запису, може виконати завдання:

1) Після введення вхідних даних та натискання на кнопку **Обчислити** його вхідні дані й відповідь відображається у списку всіх попередніх результатів.

2) Перевірити чи збігається відповідь з очікуваною.

е) Перевіряє, що користувач, який знаходиться у своєму обліковому запису, може ввести ще одні дані:

1) Переконатися, що вдалося ще одні дані.

2) Переконатися, що результат знаходиться після попереднього результату, 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 ввести [1, 2, 3].

2) В автоматичному режимі запустити ще один браузер driver2, виконати Log In від імені користувача user2 та ввести [11, 22, 33]. Протестувати, що в браузері driver2 присутні обидва коментаря – comment1 та comment2.

3) В браузері driver1 від імені користувача user1 додати дані та видалити їх. Протестувати, що в браузері driver1, присутні однакові відповіді що й в браузері driver2.

4) Закрити обидва браузера, порівняти відповіді з попередніми зареєстрованих користувачів.

ВИСНОВКИ

В даній роботі узагальнилися знання по курсу «Автоматизоване тестування програмного забезпечення». Провелося тестування веб-застосунку на модульному та приймальному рівнях. Були використані бібліотеки: unittest, subprocess, selenium, requests, tornado.

Наочно можна побачити різницю між модульним і приймальним тестуванням: якщо перший тип тестує тільки основну програми, то другий певним чином ближче до користувача і імітує його роботу. Проте перший виконується миттєво, то на другий треба значно більше часу, відповідно непередбачувана поведінка програми виявлена за допомогою модульного тестування “дешевша” і займає менше часу для виправлення та регресійного тестування.

Можна відмітити необхідність у тестуванні за допомогою http запитів, оскільки з їх допомогою можна перевірити поведінку веб-застосунку в ситуаціях, в які при звичайному тестуванні було б потрабити важко, або неможливо. Наприклад створення запису незареєстрованим користувачем, що є потенційною небезпекою.

Фінальне тестування за допомогою Selenium WebDriver дозволяло пройти шляхом користувача і віднайти непередбачувану поведінку програми на яку він міг натрапити.

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. Requests: HTTP for Humans™ [Електронний ресурс] // v2.25.1. – 2020. – Режим доступу до ресурсу: https://requests.readthedocs.io/en/master/.

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

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

5. 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.

6. 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_1.py

```
import unittest
from backend import root_mean_square


class TestTask1(unittest.TestCase):
    """Unit test task_1"""

    def test_valid_input(self):
        inp = [3, 13, 33, 87]
        self.assertAlmostEqual(root_mean_square(inp), 47)

    def test_zeros(self):
        inp = [0, 0, 0]
        self.assertAlmostEqual(root_mean_square(inp), 0)

    def test_negative(self):
        inp = [-99, -99, -1]
        self.assertAlmostEqual(root_mean_square(inp), 80.83522334)

    def test_real(self):
        inp = [0.3331, 5.032, -8989]
        self.assertAlmostEqual(root_mean_square(inp), 5189.80238647)

    def test_no_input(self):
        inp = []
        self.assertAlmostEqual(root_mean_square(inp), None)

    def test_type_error(self):
        inp = ['1', 90, 4]
        self.assertRaises(TypeError, root_mean_square, inp)


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

Лістинг файлу test_2.py

```
from subprocess import Popen, PIPE
import unittest
import requests
import pprint
from time import sleep


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

    Linus = ('Linus T.', 'torvalds@osdl.org', 'kernel')
    Iryna = ('Iryna K.', 'kostushkoia5@gmail.com', 'numericalerror')

    def setUp(self):
        self.backend_process = Popen(
            ['python', self.BACKEND_NAME],)
        sleep(2)

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

    def test_login(self):
        """
        Вхід зареєстрованого користувача
        """
        input_data = {'email': self.Linus[1], 'password': self.Linus[2]}

        response = requests.post(
            self.START_URL + '/api/v1/user/login', json=input_data)
        self.assertTrue(response.status_code == 200)

        result = response.json()
        self.assertEqual(result['display_name'], self.Linus[0])
        self.assertIn('token', result)
        self.assertIn('user_id', result)

    def test_wrong_pass(self):
        """
```

```
    Вхід з невірним паролем
    """
    input_data = {'email': self.Linus[1], 'password': 'incorrect password'}
    response = requests.post(
        self.START_URL + '/api/v1/user/login', json=input_data)
    self.assertEqual(response.status_code, 403)

def test_wrong_login(self):
    """
    Вхід з невірним логіном
    """
    input_data = {'email': 'incorrect email', 'password': self.Linus[1]}
    response = requests.post(
        self.START_URL + '/api/v1/user/login', json=input_data)
    self.assertEqual(response.status_code, 403)

def test_unauthorized_comment(self):
    """
    Створення коментаря в обхід входу
    """
    # Let's just assume we know that test Bob's user id is 2.
    required_cookies = {'jwt': 'definitely incorrect security token'}
    data = {'user_id': 2,
            "numbers": [3434, 4343]}
    response = requests.post(
        self.START_URL + '/api/v1/task',
        json=data, cookies=required_cookies)
    self.assertEqual(
        response.status_code, 401, 'Unauthorized request to add comment has actually succeeded')

def test_logout(self):
    """
    Вихід з облікового запису
    """
    # login
    input_data = {'email': self.Linus[1], 'password': self.Linus[2]}
    response = requests.post(
        self.START_URL + '/api/v1/user/login', json=input_data)
    result = response.json()
    jwt = result['token']
    # logout
    data = {'id': result['user_id'], }
    required_cookies = {'jwt': jwt}
    r = requests.post(self.START_URL + '/api/v1/user/logout',
                      json=data, cookies=required_cookies)
    self.assertEqual(
        r.status_code, 200, "User can't log out")

def test_authorized_comment(self):
    """
    Вирішення задачі з вхідними данними і створення коментаря
    """
    # login
    input_data = {'email': self.Linus[1], 'password': self.Linus[2]}
    response = requests.post(
        self.START_URL + '/api/v1/user/login', json=input_data)
    result = response.json()
    jwt = result['token']
    required_cookies = {'jwt': jwt}
    data = {'user_id': result['user_id'],
            "numbers": [3, 13, 33, 87]}
    r = requests.post(self.START_URL + '/api/v1/task',
                      json=data, cookies=required_cookies)
    expected = {"comments": [
        {
            "id": 1,
            "text": "Input: [2, 4, 16, 256] Answer: 128.2692480682724",
            "user_id": 1,
            "display_name": "Linus T."
        },
        {
            "id": 2,
            "text": "Input: [1, -1, 4, -4] Answer: 2.9154759474226504",
            "user_id": 2,
            "display_name": "Iryna K."
        },
        {
            "id": 3,
            "text": "Input: [3, 13, 33, 87] Answer: 47.0",
            "user_id": 1,
            "display_name": "Linus T."
        }
    ],
        "answer": 47.0
    }
    self.assertEqual(
        r.json(), expected, "Can't post comments")

def test_singin(self):
    """
    Реєстрація нового користувача
    """
    input_data = {"display_name": "Dmytro",
                  "password": "dima",
                  "email": "dimanavsisto@gmail.com"}
    response = requests.post(
        self.START_URL + '/api/v1/user/signup', json=input_data)
    result = response.json()
    expected = {
        "token": "aef6967d-f444-45ec-a624-47b6ec7c2c12",
        "user_id": 3,
```

```
            "display_name": "Dmytro"}
        self.assertEqual(
            result["display_name"], expected["display_name"], "User can't signin")

    def test_singin_existing(self):
        """
        Реєстрація існуючого користувача
        """
        input_data = {
            "display_name": self.Linus[0],
            "password": self.Linus[2],
            "email": self.Linus[1]}
        response = requests.post(
            self.START_URL + '/api/v1/user/signup', json=input_data)
        self.assertEqual(response.status_code, 409)


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

Лістинг файлу test_3.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'
    header_pause = 2
    def setUp(self):
        self.backend_process = Popen(['python', self.BACKEND_NAME])
        self.driver = webdriver.Firefox()
        self.maxDiff = None

    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, numbers):
        # driver.find_element(By.XPATH, '//button[.="New comment"]').click()
        for i, n in enumerate(numbers):
            elem = driver.find_element(By.CSS_SELECTOR, f"input:nth-child({i + 1})")
            elem.clear()
            elem.send_keys(str(n))
        solve = driver.find_element(By.XPATH, '//button[.="Обчислити"]')
        solve.click()

    def _check_answer(self, driver, answer):
        elem_answer = driver.find_element(By.CSS_SELECTOR, "h3")
        answer_in_elem = elem_answer.text[11:]
        self.assertEqual(answer_in_elem, answer, "Answer isn't correct")

    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)
        elems = driver.find_elements(
            By.XPATH, '//h1[.="Курсова робота"]')
        self.assertTrue(elems, 'Header is absent')

        elems = driver.find_elements(
            By.XPATH, '//h2[text()!=""]')
        self.assertTrue(elems, 'Condition body is empty')

        comments = (
            ('Input: [2, 4, 16, 256] Answer: 128.2692480682724', 'Linus T.'),
            ('Input: [1, -1, 4, -4] Answer: 2.9154759474226504', 'Iryna K.'),
        )

        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),
                             'Answers does not match')

        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')
        # Display Name
        elems = driver.find_elements(
            By.XPATH, '//input[@name="display_name"]')
        self.assertTrue(
            elems, 'There is no field for entering a Display name')
        elems = driver.find_elements(
            By.XPATH, '//div[@id="signup-section"]/div/label[1]/b')
        self.assertTrue(
            elems, 'There is no label for a Display name')
        # Email
        elems = driver.find_elements(
            By.XPATH, '//input[@name="email"]')
        self.assertTrue(
            elems, 'There is no field for entering an Email')
        elems = driver.find_elements(
            By.XPATH, '//div[@id="signup-section"]/div/label[2]/b')
        self.assertTrue(
            elems, 'There is no label for an Email')
        # Password
        elems = driver.find_elements(
            By.XPATH, '//input[@name="password"]')
        self.assertTrue(
            elems, 'There is no field for entering a Password')
        elems = driver.find_elements(
            By.XPATH, '//div[@id="signup-section"]/div/label[3]/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)
        # 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, "//div[@id='signup-section']/div/button").click()
        # Name check
        raw_text = driver.find_element(By.ID, 'signup-section').text
        self.assertEqual(
            raw_text[:-len(self.LOGOUT_SIGNATURE)-1], 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 = 'torvalds@osdl.org', 'kernel'

        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)-1], 'Linus T.')
            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 ввід завдань та кнопка Обчислити
        """
        email, password = 'torvalds@osdl.org', 'kernel'
        with self.driver as driver:
            driver.get(self.START_URL)
            self._log_in(driver, email, password)

            elem = driver.find_element(By.XPATH, "//div[@id='values']/input")
            self.assertTrue(elem, 'Input values is absent')
            elem = driver.find_element(By.XPATH, '//button[.="Обчислити"]')
            self.assertTrue(elem, 'New comment button is missing')

    def test_5_comment(self):
        """
        5. Перевіряє, що користувач, може виконати завдання
        """
        name = 'Linus T.'
        email, password = 'torvalds@osdl.org', 'kernel'
        numbers = [3, 13, 33, 87]
        my_comment = "Input: [3, 13, 33, 87] Answer: 47.0"
        with self.driver as driver:
            driver.get(self.START_URL)
            self._log_in(driver, email, password)
            # a
            self._post_comment(driver, numbers)
            self._check_answer(driver, "47")
            self._check_last_comment(driver, my_comment, name)
            # bad input
            self._post_comment(driver, [99, 'numbers', '', ''])
            self._check_answer(driver, "")

    def test_6_comment(self):
        """
        6. Перевіряє, що користувач, може ввести ще один коментар
        """
        name = 'Linus T.'
        email, password = 'torvalds@osdl.org', 'kernel'
        numbers_1 = (3, 13, 33, 87)
        numbers_2 = (-16, 16, -16, 16)
        comment_numbers = (numbers_1, numbers_2)
        comments = (
            "Input: [3, 13, 33, 87] Answer: 47.0",
            "Input: [-16, 16, -16, 16] Answer: 16.0",
        )
        with self.driver as driver:
            driver.get(self.START_URL)
            self._log_in(driver, email, password)
            for numbers in comment_numbers:
                self._post_comment(driver, numbers)
            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
        """
        name = 'Mary Jane'
        email, password = 'maryjane420@hh.org', 'canintoabyss'
        with self.driver as driver:
```

```
            driver.get(self.START_URL)
            self._create_user(driver, name, email, password)
            self._log_in(driver, email, password)
            comments_before = driver.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[1]')
            self._post_comment(driver, [11, 13, 15])
            time.sleep(0.5)
            driver.find_element(By.XPATH, '//div[@id="comments"]/ul/li[last()]/span/button[.="Remove"]').click()
            time.sleep(0.5)
            comments_after = driver.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[1]')
            c1 = list(map(lambda x: x.text, comments_before))
            c2 = list(map(lambda x: x.text, comments_after))

            self.assertEqual(c1, c2, "Коментарі 'до' і 'після' відрізняються")

    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, [1, 2, 3])
            # b
            self._log_in(driver_2, *user_2[1:])
            self._post_comment(driver_2, [11, 22, 33])
            # add and remove
            self._post_comment(driver_1, [1, 2, 3])
            driver_1.find_element(By.XPATH, '//div[@id="comments"]/ul/li[last()]/span/button[.="Remove"]').click()

            comments_user_2 = driver_2.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[1]')

            comments_user_1 = driver_1.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span[1]')
            c1 = list(map(lambda x: x.text, comments_user_1))
            c2 = list(map(lambda x: x.text, comments_user_2))
            self.assertEqual(c1, c2, "Коментарі в двох користувачів відображаються по-різному")
        with webdriver.Firefox() as driver:
            driver.get(self.START_URL)
            comments_after = driver.find_elements(
                By.XPATH,
                '//div[@id="comments"]/ul/li/span')
            c = list(map(lambda x: x.text, comments_after))
        self.assertEqual(c1, c)


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

Додаток Б

Скріншоти результатів виконання автотестів

```
(venv-gmbot) ~/K/A/L/C/source (main↑1|✚3) $ python test_1.py -v
test_negative (__main__.TestTask1) ... ok
test_no_input (__main__.TestTask1) ... ok
test_real (__main__.TestTask1) ... ok
test_type_error (__main__.TestTask1) ... ok
test_valid_input (__main__.TestTask1) ... ok
test_zeros (__main__.TestTask1) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.000s

OK
```

Рисунок Б.1 – Скріншот тестування test_1.py

```
(venv-gmbot) ~/K/A/L/C/source (main↑1|✚3) $ python test_2.py -v                    18:27:47
test_authorized_comment (__main__.AppTest)
Вирішення задачі з вхідними данними і створення коментаря ... ok
test_login (__main__.AppTest)
Вхід зареєстрованого користувача ... ok
test_logout (__main__.AppTest)
Вихід з облікового запису ... ok
test_singin (__main__.AppTest)
Реєстрація нового користувача ... ok
test_singin_existing (__main__.AppTest)
Реєстрація існуючого користувача ... WARNING:tornado.access:409 POST /api/v1/user/signup (::1) 0.33ms
ok
test_unauthorized_comment (__main__.AppTest)
Створення коментаря в обхід входу ... WARNING:tornado.access:401 POST /api/v1/task (::1) 1.55ms
ok
test_wrong_login (__main__.AppTest)
Вхід з невірним логіном ... WARNING:tornado.access:403 POST /api/v1/user/login (::1) 0.35ms
ok
test_wrong_pass (__main__.AppTest)
Вхід з невірним паролем ... WARNING:tornado.access:403 POST /api/v1/user/login (::1) 0.33ms
ok

----------------------------------------------------------------------
Ran 8 tests in 16.148s

OK
```

Рисунок Б.2 – Скріншот тестування test_2.py

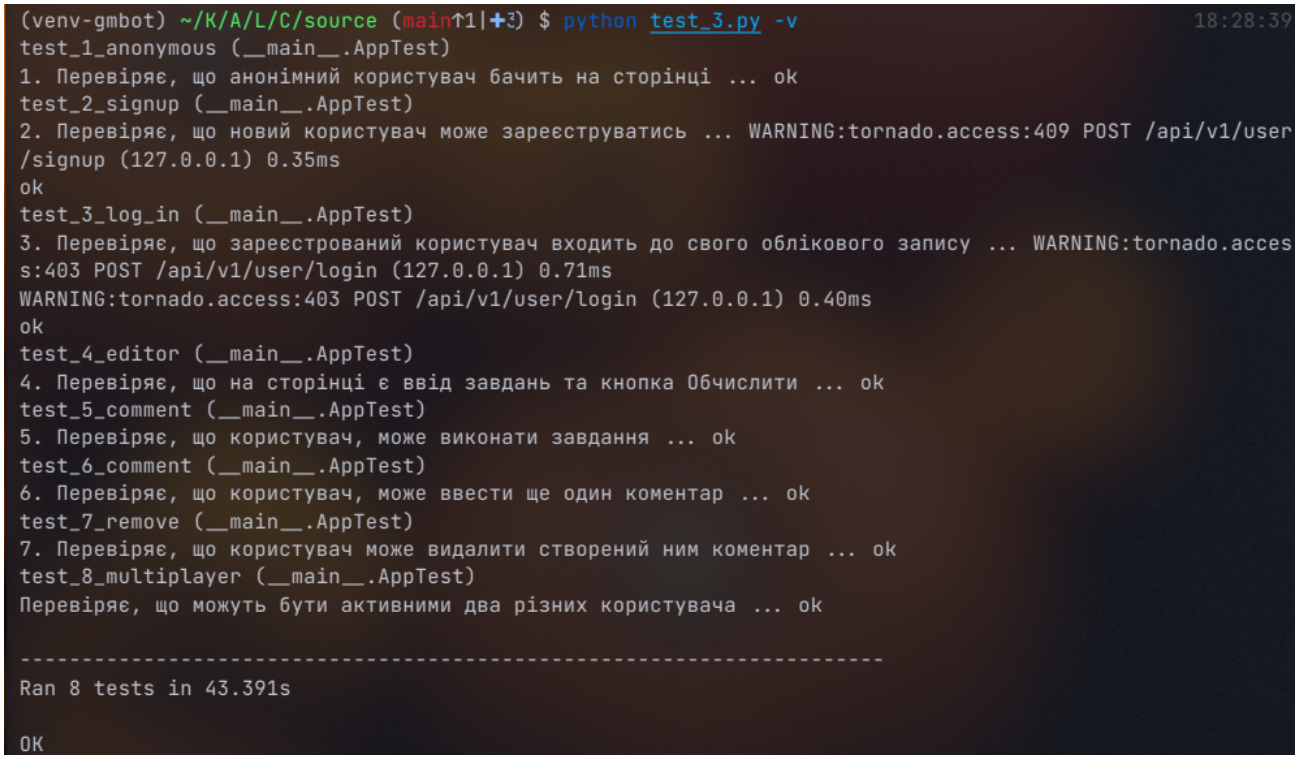

Рисунок Б.3 – Скріншот тестування test_3.py