Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

Лабораторна робота №5

Тема: «Написання тестів для користувацьких сценаріїв»

Виконала

ст. гр. ІП-21-3

Лаба Ю.С.

Перевірив

Храбатин Р.І.

м. Івано-Франківськ

2025

**Мета роботи:** навчитися створювати тести для перевірки користувацьких сценаріїв, розібратися в існуючих інструментах для створення тестів для перевірки користувацьких сценаріїв та навчитися їх обирати.

**Короткі теоретичні відомості**

Наскрізне тестування (англ. *End-to-End testing*, E2E) — це тип функціонального тестування, при якому перевіряється повна функціональність додатку з точки зору кінцевого користувача. Мета — змоделювати реальну поведінку користувача та перевірити, чи всі компоненти системи правильно взаємодіють між собою.

На відміну від юніт-тестів, які перевіряють окремі функції чи модулі, E2E-тести проходять через весь стек застосунку — від інтерфейсу користувача до бази даних, включаючи серверну логіку, API, аутентифікацію та інші критичні компоненти.

Мета E2E тестування:

* Перевірити логіку користувацьких сценаріїв: наприклад, реєстрація, логін, додавання товару в кошик, оформлення замовлення.
* Переконатися, що всі частини системи працюють разом належним чином.
* Виявити помилки на інтеграційному рівні, які неможливо знайти за допомогою юніт- або інтеграційних тестів.
* Гарантувати якість продукту з точки зору UX (user experience).

Одним із популярних інструментів для написання E2E тестів у середовищі JavaScript є Puppeteer — бібліотека, яка забезпечує високорівневий API для керування браузером Chrome або Chromium через протокол DevTools.

Переваги Puppeteer:

* Повний контроль над браузером: відкриття сторінок, введення тексту, кліки, скріншоти тощо.
* Підтримка перевірки в headless-режимі (без графічного інтерфейсу).
* Добре підходить для UI/E2E тестування, web scraping та PDF генерації.
* Легко інтегрується з Jest, Mocha або іншими фреймворками.

Основні поняття:

* Headless browser - браузер без графічного інтерфейсу. Використовується для швидшого автоматичного тестування.
* Selector - CSS-селектор елемента DOM, який використовується для взаємодії (наприклад, кнопка або поле вводу).
* Assertion - Твердження, що перевіряє, чи результат відповідає очікуванням. У Jest це expect(value).toBe(...).
* waitForSelector() - функція, яка чекає появи елемента на сторінці перед взаємодією з ним.
* Networkidle - стан, коли всі мережеві запити завершено. Корисно при очікуванні завантаження сторінки.

Ключові принципи ефективного E2E тестування:

1. Орієнтація на користувача — тести мають імітувати дії реальних користувачів.
2. Повна інтеграція — охоплення як frontend, так і backend логіки.
3. Використання динамічного очікування, а не фіксованих таймаутів.
4. Модульність тестів — розбиття великих сценаріїв на підфункції для повторного використання.
5. Читабельність — іменування функцій і опис дій мають бути зрозумілими іншим розробникам або тестувальникам.

**Хід роботи**

**Завдання перше:** Оберіть ресурс який буде протестовано

Сайт <https://demoqa.com> — це безкоштовний ресурс для практики автоматизованого тестування вебінтерфейсів. Він містить різноманітні інтерактивні елементи: форми, таблиці, алерти, драг-н-дроп, радіокнопки, чекбокси та інші компоненти. Цей сайт часто використовується для навчання та перевірки роботи end-to-end тестів із такими інструментами, як Puppeteer, Cypress або Selenium.

**Завдання друге:** Написати реалізацію трьох користувацьких сценарії до обраного ресурсу. Прохання, коротко перед реалізацією опишіть, що ви перевірятимете даним тестом. Це можна зробити у вигляді списку, як подано вище, оскільки в реалізації буде видно як саме ви проводите саме тестування.

**1. Radio Buttons**

* Відкриваємо сторінку з радіокнопками: <https://demoqa.com/radio-button>.
* Очікуємо на появу елемента "Yes" та клікаємо на нього.
* Чекаємо, поки з’явиться елемент із класом .text-success.
* Перевіряємо, що його текст дорівнює "Yes" — радіокнопка була обрана успішно.

test('Radio Buttons - should select "Yes" and check response', **async** () **=>** {

    await page.goto('https://demoqa.com/radio-button', { waitUntil: 'networkidle2' });

    await page.waitForSelector('label[for="yesRadio"]', { visible: true });

    await page.click('label[for="yesRadio"]');

    await page.waitForSelector('.text-success', { visible: true });

**const** result = await page.$eval('.text-success', el **=>** el.textContent.trim());

    expect(result).toBe('Yes');

  });

**2. Web Tables**

* Переходимо на сторінку з таблицями: <https://demoqa.com/webtables>.
* Чекаємо на кнопку Add, натискаємо її.
* Очікуємо на появу модальної форми (поле #firstName).
* Перевіряємо, що модальне вікно дійсно видиме — форма успішно з’явилась.

test('Web Tables - should click Add button and see modal form', **async** () **=>** {

    await page.goto('https://demoqa.com/webtables', { waitUntil: 'networkidle2' });

    await page.waitForSelector('#addNewRecordButton', { visible: true });

    await page.click('#addNewRecordButton');

    await page.waitForSelector('#firstName', { visible: true });

**const** formVisible = await page.$eval('#registration-form-modal', el **=>** getComputedStyle(el).display !== 'none');

    expect(formVisible).toBe(true);

  });

**3. Drag and Drop**

* Відкриваємо сторінку з перетягуванням: <https://demoqa.com/droppable>.
* Знаходимо обидва елементи: той, який треба перетягнути (#draggable) і куди перетягнути (#droppable).
* Емулюємо мишку: натискаємо, перетягуємо в ціль і відпускаємо.
* Очікуємо на зміну тексту у приймаючому блоці на "Dropped!" — перетягування спрацювало.

test('Drag and Drop - should move element from left to right',

**async** () **=>** {

      await page.goto('https://demoqa.com/droppable', { waitUntil: 'networkidle2' });

**const** draggable = await page.$('#draggable');

**const** droppable = await page.$('#droppable');

**const** draggableBox = await draggable.boundingBox();

**const** droppableBox = await droppable.boundingBox();

      await page.mouse.move(

        draggableBox.x + draggableBox.width / 2,

        draggableBox.y + draggableBox.height / 2

      );

      await page.mouse.down();

      await page.mouse.move(

        droppableBox.x + droppableBox.width / 2,

        droppableBox.y + droppableBox.height / 2,

        { steps: 20 }

      );

      await page.mouse.up();

      await page.waitForFunction(

        () **=>** document.querySelector('#droppable p').textContent.includes('Dropped!'),

        { timeout: 5000 }

      );

    },

    15000

  );

**4. Dynamic Properties**

* Переходимо на сторінку з динамічними властивостями: <https://demoqa.com/dynamic-properties>.
* Очікуємо, поки кнопка #enableAfter стане доступною для кліку (enabled).
* Перевіряємо, що вона справді активна — динамічна активація працює.

test('Dynamic Properties - should wait for enabled button',

**async** () **=>** {

      await page.goto('https://demoqa.com/dynamic-properties', { waitUntil: 'domcontentloaded' });

      await page.waitForSelector('#enableAfter:enabled', { timeout: 10000 });

**const** isEnabled = await page.$eval('#enableAfter', el **=>** !el.disabled);

      expect(isEnabled).toBe(true);

    },

    15000

  );

* 1. **Modal Dialog**
* Відкриваємо сторінку з модальними вікнами: <https://demoqa.com/modal-dialogs>.
* Натискаємо кнопку Show Small Modal.
* Очікуємо появу заголовку Small Modal.
* Перевіряємо, що вікно відкрилось правильно.
* Натискаємо кнопку Close, щоб закрити його.
* Перевіряємо, що вікно успішно зникло.

test('Modal Dialog - should open and close small modal',

**async** () **=>** {

      await page.goto('https://demoqa.com/modal-dialogs', { waitUntil: 'domcontentloaded' });

      await page.click('#showSmallModal');

      await page.waitForSelector('#example-modal-sizes-title-sm', { visible: true });

**const** modalTitle = await page.$eval('#example-modal-sizes-title-sm', el **=>** el.textContent);

      expect(modalTitle).toBe('Small Modal');

      await page.click('#closeSmallModal');

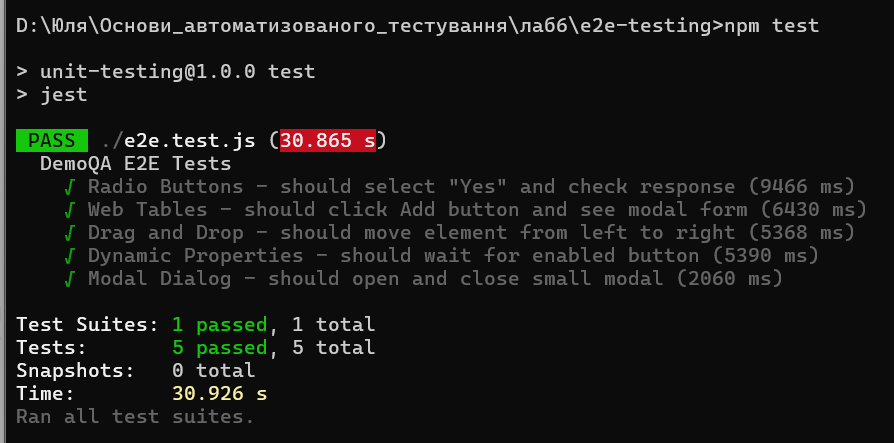
      await page.waitForSelector('#example-modal-sizes-title-sm', { hidden: true });

    },

    15000

  );

Результат:

****

**Висновок:** У результаті виконання лабораторної роботи було створено серію наскрізних E2E-тестів за допомогою бібліотеки Puppeteer для сайту <https://demoqa.com>. Ці тести охоплюють базові сценарії взаємодії з користувацьким інтерфейсом: вибір радіокнопок, відкриття модального вікна, перетягування елементів, очікування динамічних змін на сторінці.

**Посилання на репозиторій:** <https://github.com/yuliia-laba/lab6>