

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет
нафти і газу

Кафедра інженерії програмного забезпечення

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни

Основи автоматизованого тестування

Виконав: ст. гр. ІІ-23-1К
Гнатюк Д. М.

Івано-Франківськ

2025

Тема: Написання тестів для графічного інтерфейсу

Мета: Навчитися створювати UI тести, розібратися в існуючих інструментах для створення UI тестів та навчитися їх обирати.

Обраний інструмент:

Puppeteer — Node.js бібліотека для керування Chrome/Chromium через DevTools Protocol. Дозволяє створювати e2e та UI тести.

Обраний сайт:

<https://magento.softwaretestingboard.com/men/tops-men/jackets-men.html>

Написані кейси:

1. Перевірка заголовка сторінки.
2. Перевірка, що товари на сторінці присутні.
3. Перевірка наявності фільтрів.
4. Перевірка, що товарів рівно 11.
5. Перевірка сортування за ціною.

```
const puppeteer = require("puppeteer");

let browser;
let page;
const URL = "https://magento.softwaretestingboard.com/men/tops-men/jackets-men.html";

beforeAll(async () => {
  browser = await puppeteer.launch({ headless: true });
  page = await browser.newPage();
});

afterAll(async () => {
  await browser.close();
});

beforeEach(async () => {
  await page.goto(URL, { waitUntil: "domcontentloaded" });
});

test("Перевірка заголовка сторінки", async () => {
  const title = await page.title();
  expect(title).toMatch(/Jackets/i);
});

test("Перевірка наявності товарів на сторінці", async () => {
  const items = await page.$$eval(".product-item", items => items.length);
  expect(items).toBeGreaterThan(0);
});

test("Перевірка наявності фільтрів", async () => {
  const filters = await page.$$eval(".filter-options-title", items => items.length);
  expect(filters).toBeGreaterThan(0);
});
```

```

});

test("Перевірка кількості товарів дорівнює 13", async () => {
  const items = await page.$$eval(".product-item", items => items.length);
  expect(items).toBe(13);
});

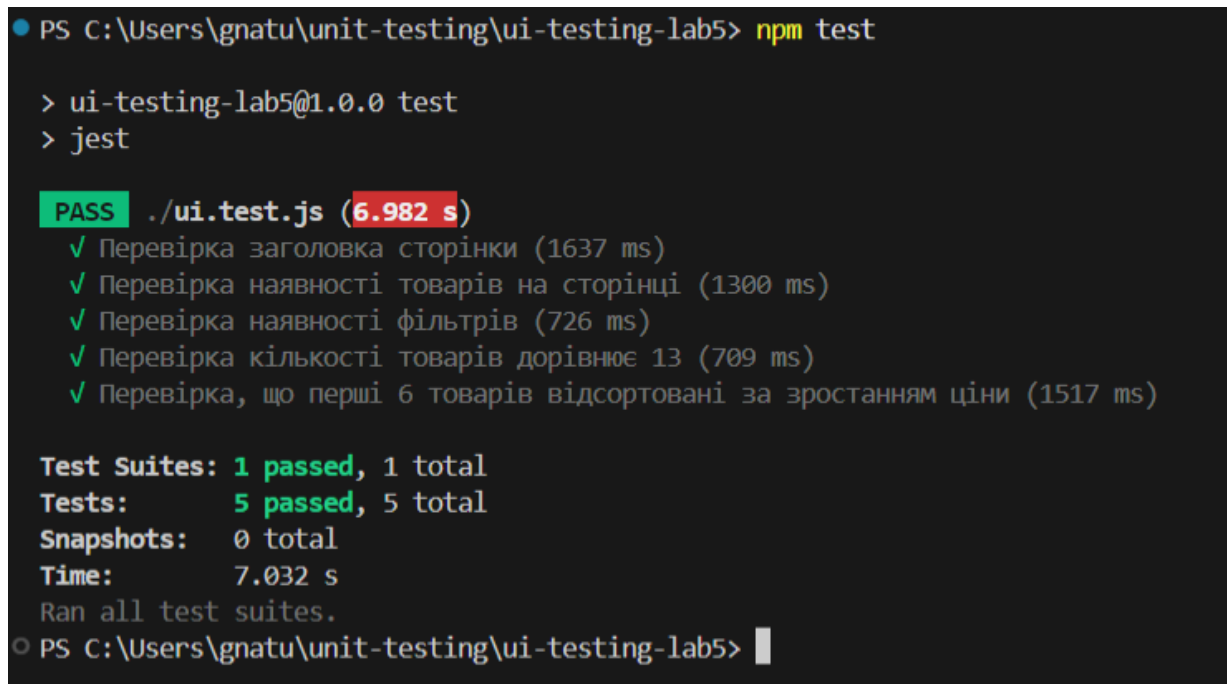
test("Перевірка сортування за ціною", async () => {
  await page.waitForSelector("select.sorter-options");
  await page.select("select.sorter-options", "price");
  await page.waitForTimeout(2000);

  const prices = await page.$$eval(".price", nodes =>
    nodes.map(n => parseFloat(n.textContent.replace("$", "").trim()))
    .filter(n => !isNaN(n))
  );

  const sorted = [...prices].sort((a, b) => a - b);
  expect(prices).toEqual(sorted);
});

```

Результат виконання тестів:



```

PS C:\Users\gnatu\unit-testing\ui-testing-lab5> npm test

> ui-testing-lab5@1.0.0 test
> jest

PASS ./ui.test.js (6.982 s)
  ✓ Перевірка заголовка сторінки (1637 ms)
  ✓ Перевірка наявності товарів на сторінці (1300 ms)
  ✓ Перевірка наявності фільтрів (726 ms)
  ✓ Перевірка кількості товарів дорівнює 13 (709 ms)
  ✓ Перевірка, що перші 6 товарів відсортовані за зростанням ціни (1517 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        7.032 s
Ran all test suites.
PS C:\Users\gnatu\unit-testing\ui-testing-lab5>

```

Висновок: У ході виконання лабораторної роботи було розглянуто підходи до автоматизованого тестування графічного інтерфейсу користувача (UI) вебзастосунку за допомогою інструменту Puppeteer у зв'язці з фреймворком Jest. Автоматизація UI-тестування дозволяє значно підвищити надійність, зменшити кількість ручних перевірок і пришвидшити виявлення помилок на етапах розробки. Здобуті знання та навички

можуть бути застосовані в реальних проектах для перевірки коректності роботи вебінтерфейсів.

Відповіді на запитання

1. Що таке UI тестування?

UI тестування (User Interface Testing) — це процес перевірки того, що елементи графічного інтерфейсу користувача (кнопки, поля, форми тощо) працюють правильно, виглядають коректно та забезпечують очікувану поведінку при взаємодії з користувачем.

2. В чому полягає суть проведення UI тестування?

Суть UI тестування полягає в імітації дій користувача над вебінтерфейсом (наприклад, кліки, введення тексту, навігація) та перевірці того, що результат цих дій відповідає очікуванням. Це дозволяє виявити помилки у візуальному відображенні та логіці роботи елементів до релізу продукту.

3. Які інструменти створення автоматизованих UI тестів вам відомі?

Серед популярних інструментів автоматизації UI тестування:

- Puppeteer – Node.js бібліотека для керування Chromium.
- Cypress – фреймворк для E2E тестування в браузері.
- Playwright – кросбраузерний інструмент від Microsoft.
- Selenium WebDriver – класичний фреймворк для UI тестування в різних мовах.

4. За допомогою якої команди можна отримати всі елементи із зазначеним селектором?

У Puppeteer:

```
const elements = await page.$$('селектор');
```

або для обробки в масиві:

```
const result = await page.$$eval('селектор', nodes => /* callback */);
```

5. За допомогою якої команди можна модифікувати отриманий результат із зазначеним селектором?

```
await page.$$eval('селектор', nodes => {  
  return nodes.map(node => node.textContent.trim());  
});
```

Це дозволяє пройтись по кожному елементу та отримати або змінити його властивості.

6. За допомогою якої команди можна перейти на певну сторінку?

```
await page.goto("https://example.com", { waitUntil: "domcontentloaded" });
```

Ця команда відкриває вказану сторінку у вкладці браузера.

7. За допомогою якої команди можна підрахувати кількість елементів?

```
const count = await page.$$eval(селектор, nodes => nodes.length);
```

Це дозволяє отримати кількість елементів, які відповідають селектору.