МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет
нафти і газу

Кафедра інженерії програмного забезпечення

ЛАБОРАТОРНА РОБОТА № 7

з дисципліни

Основи автоматизованого тестування

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

Івано-Франківськ

2025

Тема: Побудова базового автоматизаційного рішення

Мета: Зрозуміти та навчитися будувати прості базові автоматизовані рішення.

Github: https://github.com/DementialDima/Lab1-AutomatedTesting

Було створено автоматизоване тестове рішення для сайту https://magento.softwaretestingboard.com з використанням Node.js, Puppeteer, Jest та Axios.

Розроблена структура включає окремі каталоги для різних рівнів тестування:

- Unit тести (`tests/unit`) для перевірки окремих функцій, наприклад, парсингу цін;

- UI тести (`tests/ui`) для перевірки інтерфейсу користувача, таких як логотип, пошук, футер;

- API тести (`tests/api`) для взаємодії з публічним HTTP API сайту;

- E2E тести (`tests/e2e`) з повною симуляцією сценаріїв користувача (реєстрація, пошук, додавання в корзину).

Під час розробки тестів було дотримано принципів DRY, KISS, POM.

Код Unit тестів:

```
const parsePrice = str => parseFloat(str.replace('$', '').replace(',', ''));

test('Парсинг ціни', () => {
  expect(parsePrice('$1,299.99')).toBe(1299.99);
});

test('Парсинг нуля', () => {
  expect(parsePrice('$0')).toBe(0);
});

test('Парсинг дробової ціни', () => {
  expect(parsePrice('$59.95')).toBe(59.95);
});

test('Парсинг з пробілами', () => {
  expect(parsePrice('  $99  ')).toBe(99);
});

test('Неправильний ввід', () => {
  expect(isNaN(parsePrice('not-a-price'))).toBe(true);
});
```

Код UI тестів:

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

let browser, page;

beforeAll(async () => {
  browser = await puppeteer.launch();
  page = await browser.newPage();
  await page.goto('https://magento.softwaretestingboard.com');
});

afterAll(() => browser.close());
```

```
test('Перевірка логотипу', async () => {
  const logo = await page.$('.logo');
  expect(logo).toBeTruthy();
});

test('Перевірка наявності пошуку', async () => {
  const search = await page.$('#search');
  expect(search).toBeTruthy();
});

test('Перевірка наявності навігації', async () => {
  const nav = await page.$('.navigation');
  expect(nav).toBeTruthy();
});

test('Перевірка наявності футера', async () => {
  const footer = await page.$('footer');
  expect(footer).toBeTruthy();
});

test('Перевірка відображення корзини', async () => {
  const cart = await page.$('.showcart');
  expect(cart).toBeTruthy();
});
```

Код API тестів:

```
const axios = require('axios');

test('Отримання головної сторінки', async () => {
  const res = await axios.get('https://magento.softwaretestingboard.com');
  expect(res.status).toBe(200);
});

test('Перевірка контенту сторінки', async () => {
  const res = await axios.get('https://magento.softwaretestingboard.com');
  expect(res.data).toContain('Magento');
});

test('404 помилка', async () => {
  try {
    await axios.get('https://magento.softwaretestingboard.com/non-existent');
  } catch (e) {
    expect(e.response.status).toBe(404);
  }
});

test('Час відповіді менше 2 секунд', async () => {
  const start = Date.now();
  await axios.get('https://magento.softwaretestingboard.com');
  const duration = Date.now() - start;
  expect(duration).toBeLessThan(2000);
});

test('Відповідь у форматі HTML', async () => {
  const res = await axios.get('https://magento.softwaretestingboard.com');
  expect(res.headers['content-type']).toContain('text/html');
});
```

Код E2E тестів:

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

let browser, page;

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

afterAll(() => browser.close());

test('Реєстрація користувача', async () => {
                                                                                await
page.goto('https://magento.softwaretestingboard.com/customer/account/create/');
  await page.type('#firstname', 'Іван');
  await page.type('#lastname', 'Тестовий');
  const email = `test${Date.now()}@example.com`;
  await page.type('#email_address', email);
  await page.type('#password', 'Test1234!');
  await page.type('#password-confirmation', 'Test1234!');
  await page.click('button[title="Create an Account"]');
  await page.waitForSelector('.page-title');
  const title = await page.$eval('.page-title', el => el.textContent);
  expect(title).toMatch(/My Account/i);
}, 15000);

test('Пошук товару', async () => {
  await page.goto('https://magento.softwaretestingboard.com');
  await page.click('.search');
  await page.type('#search', 'jacket');
await Promise.all([
  page.waitForNavigation({ waitUntil: "domcontentloaded" }),
  page.keyboard.press("Enter")
]);
await page.waitForSelector(".product-item");
  const results = await page.$$eval('.product-item', items => items.length);
  expect(results).toBeGreaterThan(0);
}, 10000);

test('Додавання товару до корзини', async () => {
   await  page.goto('https://magento.softwaretestingboard.com/men/tops-men/jackets-
men.html');
  await page.waitForSelector('.product-item .product-item-link');
  await page.click('.product-item .product-item-link');
  await page.waitForSelector('#option-label-size-143-item-166');
  await page.click('#option-label-size-143-item-166');
  await page.click('#option-label-color-93-item-50');
  await page.click('#product-addtocart-button');
  await page.waitForSelector('.message-success');
  const message = await page.$eval('.message-success', el => el.textContent);
  expect(message).toMatch(/You added/i);
}, 15000);
```

Результат виконання тестів:

Висновок: У ході виконання лабораторної роботи було реалізовано повноцінне автоматизаційне рішення з підтримкою чотирьох рівнів тестування. Це дозволило не лише перевірити окремі компоненти додатку, але й забезпечити цілісну перевірку бізнес-сценаріїв. Використання Puppeteer для UI та E2E, а також Jest і Axios для unit та API тестів дозволило створити гнучку та розширювану архітектуру.

Відповіді на запитання

1. Що таке автоматизаційне рішення?

Автоматизаційне рішення — це структурований програмний фреймворк або система, яка дозволяє автоматизувати процес тестування програмного забезпечення, охоплюючи різні рівні: unit, UI, API, E2E. Воно включає архітектуру, скрипти, бібліотеки,

інструменти та налаштування, необхідні для ефективного, стабільного та масштабованого тестування.

2. З яких шарів може складатися автоматизаційне рішення?

- Ядро (core): конфігурації, логери, хелпери, налаштування запуску.
- Бізнес-рівень (business): логіка Page Object, API-контракти, функціональні компоненти.
- Тестовий рівень (tests): власне тести, тестові дані.
- Сервісний рівень (service): налаштування CI/CD, Git, лінтери, документація.

3. Якими принципами варто послуговуватися підчас розробки автоматизаційних рішень?

- SOLID — принципи об'єктно-орієнтованого дизайну.
- DRY (Don't Repeat Yourself) — уникнення дублювання коду.
- KISS (Keep It Simple, Stupid) — простота та зрозумілість рішень.
- POM (Page Object Model) — шаблон проєктування для UI автоматизації.

4. Скільки існує атрибутів якості по відношенню до автоматизаційних рішень?

Зазвичай виділяють 7 основних атрибутів якості:

1. Підтримуваність
2. Продуктивність
3. Легкість навчання
4. Модульність
5. Перевикористання
6. Надійність
7. Масштабованість

5. В чому полягає суть атрибуту підтримуваності?

Підтримуваність означає, що автоматизовані тести легко адаптувати до змін у продукті без значних витрат часу. Це досягається модульністю, абстракцією, централізацією змінних та ізоляцією логіки.

6. В чому полягає суть атрибуту продуктивності?

Продуктивність означає оптимальне використання ресурсів під час виконання тестів, мінімізацію часу запуску та уникнення надмірного очікування (наприклад, за допомогою динамічних, а не статичних wait-ів).

7. В чому полягає суть атрибуту легкості навчання?

Це здатність нових учасників команди швидко зрозуміти структуру проєкту, логіку тестів та принципи написання. Досягається через документацію, стандартизацію та зрозумілу архітектуру.

8. В чому полягає суть атрибуту модульності?

Модульність передбачає поділ архітектури на ізольовані компоненти, які виконують окремі завдання. Це дозволяє змінювати або повторно використовувати частини коду без впливу на решту системи.

9. В чому полягає суть атрибуту перевикористання?

Перевикористання означає можливість повторного використання функцій, компонентів і модулів у різних тестах або проектах. Це зменшує кількість дублювання та спрощує масштабування рішення.

10. В чому полягає суть атрибуту надійності?

Надійність — це здатність тестів стабільно виконуватись без хибних спрацьовувань, з однаковими результатами. Забезпечується стійкістю до змін, обробкою помилок та якісним хендлінгом очікувань.

11. В чому полягає суть атрибуту масштабованості?

Масштабованість означає, що рішення легко адаптується до більших обсягів тестів, даних або змін продукту. Воно повинно підтримувати паралельний запуск, CI/CD, розширення структури без її перебудови.