Міністерство освіти й науки України

Івано-Франківський національний технічний

університет нафти і газу

Кафедри ІПЗ

Лабораторна робота №6

# Тема: «НАПИСАННЯ ТЕСТІВ ДЛЯ КОРИСТУВАЦЬКИХ СЦЕНАРІЇВ»

Виконав студент

Групи ІП-21-2

Кавецький О.О

Перевірив

Храбатин Р. І.

м. Івано-Франківськ

2025р.

**Мета:** 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:**

**End-to-End (E2E) Тести** — це тип тестування програмного забезпечення, який перевіряє роботу додатка або системи від початку до кінця, вражаючи всю логіку, інтеграцію компонентів і функціональність в реальних умовах користувача.

### Основні аспекти E2E тестів:

1. **Мета**: Основна мета E2E тестів — перевірити, чи працює система як єдине ціле. Це включає в себе взаємодію з інтерфейсом користувача, підключення до бази даних, відправку запитів до серверу, перевірку результатів на екран та перевірку правильності обробки даних.
2. **Інструменти**:
   * Для E2E тестів часто використовуються інструменти автоматизації, які імітують поведінку користувача в браузері або мобільному додатку. Приклади таких інструментів:
     + **Puppeteer**: автоматизація браузера Google Chrome.
     + **Cypress**: сучасний інструмент для тестування інтерфейсів користувача.
     + **Selenium**: широко використовуваний інструмент для тестування веб-додатків.
     + **Playwright**: інструмент для тестування багатоплатформених додатків.
3. **Підходи**:
   * **Тестування користувацьких сценаріїв**: Тестування конкретних сценаріїв, які можуть виконувати кінцеві користувачі, наприклад, реєстрація, пошук товару, оформлення замовлення тощо.
   * **Перевірка інтеграції компонентів**: E2E тести забезпечують перевірку роботи всіх компонентів додатка в реальних умовах.
   * **Автоматизація**: Оскільки E2E тести включають великі обсяги роботи, їх часто автоматизують, щоб прискорити тестування і зменшити кількість людських помилок.
4. **Переваги E2E тестів**:
   * **Тестує весь процес**: E2E тести охоплюють усі етапи, від взаємодії користувача до серверних операцій, що дозволяє гарантувати, що всі частини системи працюють разом.
   * **Мінімізація ризиків**: Виявлення проблем на етапі інтеграції дозволяє значно знизити ймовірність помилок в реальних користувацьких сценаріях.
   * **Покриття реальних випадків**: Тести в репрезентативних умовах дозволяють перевірити додаток в умовах, схожих на реальні.
5. **Обмеження E2E тестів**:
   * **Часові витрати**: E2E тести часто займають більше часу через великі обсяги перевірок.
   * **Низька швидкість виконання**: Тестування інтерфейсів і взаємодія з базою даних можуть бути повільнішими порівняно з юніт-тестами.
   * **Важкість налагодження**: Коли тест не проходить, знайти причину помилки може бути складніше через складність та масштабність тестованого процесу.

### Ключові етапи написання E2E тестів:

1. **Налаштування середовища**: Встановлення інструментів тестування та конфігурація середовища.
2. **Написання тестів**: Створення сценаріїв, що імітують поведінку користувача в системі.
3. **Виконання тестів**: Запуск тестів та перевірка результатів.
4. **Аналіз результатів**: Перевірка успішності тестів і виявлення проблем.
5. **Оптимізація**: Після запуску тестів можуть бути внесені зміни для покращення продуктивності чи точності перевірок.

Таким чином, E2E тести є важливим інструментом для забезпечення стабільності і якості програмного забезпечення на всіх етапах його використання.

**Виконання:**

Використовуючи API з попередньої роботи напишу наступні e2e тести:

* **Тест "User Registration"**:
* Цей тест перевіряє процес реєстрації користувача на сайті. Він автоматично заповнює поля форми реєстрації випадковими даними за допомогою бібліотеки faker. Після введення імені, прізвища, електронної пошти та пароля, тест натискає кнопку для створення акаунту. Після цього перевіряється, чи відображається повідомлення про успішну реєстрацію з текстом "Thank you for registering", що підтверджує успішну реєстрацію.

  test('User Registration', async () => {

    await page.goto('https://magento.softwaretestingboard.com/customer/account/create/');

    await page.type('#firstname', faker.person.firstName());

    await page.type('#lastname', faker.person.lastName());

    const email = faker.internet.email();

    await page.type('#email\_address', email);

    const password = faker.internet.password(10);

    await page.type('#password', password);

    await page.type('#password-confirmation', password);

    await Promise.all([

      page.waitForNavigation(),

      page.click('button[title="Create an Account"]')

    ]);

    const successMessage = await page.$eval('.message-success div', el => el.textContent);

    expect(successMessage).toMatch(/Thank you for registering/);

  }, 20000);

* **Тест "Search Product"**:
* Цей тест перевіряє функціональність пошуку продуктів на сайті. Він вводить запит "jacket" у поле пошуку, натискає Enter і чекає, поки з'являться результати пошуку. Тест перевіряє, чи знайдено хоча б один продукт, що підтверджує правильність роботи пошукової системи на сайті.

  test('Search Product', async () => {

    await page.goto('https://magento.softwaretestingboard.com/');

    await page.type('#search', 'jacket');

    await page.keyboard.press('Enter');

    await page.waitForSelector('.product-item');

    const products = await page.$$('.product-item');

    expect(products.length).toBeGreaterThan(0);

  }, 15000);

* **Тест "Add to Cart and Proceed to Checkout"**:
* Цей тест перевіряє можливість додавання продукту до кошика та перехід до оформлення замовлення. Тест виконується наступним чином:
  + Перше, він переходит на головну сторінку сайту і чекає, поки відобразиться перший продукт.
  + Потім клікає на продукт для перегляду детальної сторінки товару, де вибираються опції розміру та кольору, якщо вони доступні.
  + Після вибору опцій продукт додається до кошика.
  + Після цього тест перевіряє, чи з'явилося повідомлення про успішне додавання товару в кошик.
  + Далі він переходить до сторінки кошика і перевіряє, чи є кнопка для переходу до оформлення замовлення.

  test('Add to Cart and Proceed to Checkout', async () => {

    await page.goto('https://magento.softwaretestingboard.com/');

    await page.waitForSelector('.product-item-info a.product-item-photo', { timeout: 10000 });

    const productLink = await page.$('.product-item-info a.product-item-photo');

    if (!productLink) {

      throw new Error('Не знайдено товар для додавання в кошик.');

    }

    await productLink.click();

    await page.waitForSelector('.swatch-attribute.size .swatch-option', { timeout: 8000 });

    await page.waitForSelector('.swatch-attribute.color .swatch-option', { timeout: 8000 });

    const sizeOption = await page.$('.swatch-attribute.size .swatch-option');

    if (sizeOption) {

      await sizeOption.click();

      await page.waitForSelector('.swatch-attribute.size .swatch-option.selected', { timeout: 5000 });

    }

    const colorOption = await page.$('.swatch-attribute.color .swatch-option');

    if (colorOption) {

      await colorOption.click();

      await page.waitForSelector('.swatch-attribute.color .swatch-option.selected', { timeout: 5000 });

    }

    await page.click('#product-addtocart-button');

    await page.waitForSelector('.message-success', { timeout: 10000 });

    await page.goto('https://magento.softwaretestingboard.com/checkout/cart/');

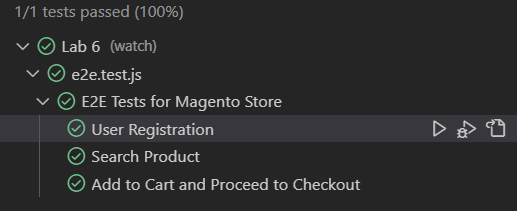
    await page.waitForSelector('.checkout-methods-items button', { timeout: 10000 });

    const checkoutBtn = await page.$('.checkout-methods-items button');

    expect(checkoutBtn).not.toBeNull();

  }, 30000);

**Результат виконання:**

****

**Висновок:** В даній роботі ми навчилися створювати тести для перевірки користувацьких сценаріїв, розібралися в існуючих інструментах для створення тестів для перевірки користувацьких сценаріїв та навчилися їх обирати.

Посилання на репозиторій - <https://github.com/olehkavetskyi/FAT-Labs>