МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

МЕТОДИЧНІ ВКАЗІВКИ

до виконання
комплексного практичного індивідуального завдання
з дисципліни

«ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ»

здобувачів ступеня вищої освіти «бакалавр»
спеціальностей 121 «Інженерія програмного забезпечення»
126 «Інформаційні системи та технології»

ТЕРНОПІЛЬ
ЗУНУ
2023

Методичні вказівки до виконання комплексного практичного індивідуального завдання з дисципліни «Якість програмного забезпечення та тестування» здобувачів ступеня вищої освіти «бакалавр» спеціальностей 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології»

Укладачі:
Тимчишин В.С., викладач кафедри комп'ютерних наук
Крепич С.Я., к.т.н., доцент кафедри комп'ютерних наук
Співак І.Я., к.т.н., доцент кафедри комп'ютерних наук
Мельник А.М., д.т.н., доцент кафедри комп'ютерних наук
Манжула В.І., к.т.н., доцент кафедри комп'ютерних наук

Відповідальний за випуск: д.т.н., професор Пукас А.В.

Методичні вказівки розглянуто та рекомендовано до друку
на засіданні кафедри комп'ютерних наук
(протокол №15 від 6 червня 2023 року)

Методичні вказівки містять основні вимоги щодо організації, змісту та виконання комплексного практичного індивідуального завдання

ЗМІСТ

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Тестування

Тестування програмного забезпечення – це техніка контролю якості, що перевіряє відповідність між реальною і очікуваною поведінкою програми завдяки кінцевому набору тестів, які обираються певним чином.

Якість не є абсолютною, це суб'єктивне поняття. Тому тестування, як процес своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення й забезпечення якості програмного забезпечення, до якого належать всі складові тестування.

Зазвичай, поняття якості обмежується такими поняттями як коректність, надійність, практичність, безпечність, але може містити більше технічних вимог, котрі описані у стандарті ISO 9126. Склад та зміст супутньої документації процесу тестування визначається стандартом IEEE 829–1998 Standard for Software Test Documentation. Існує багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів – це по суті дослідницький та творчий процес, а не тільки створення та виконання рутинної процедури.

У відповідність з процесами або методологіями розробки ПЗ, під час проведення тестування створюється і використовується певна кількість тестових артефактів (документи, моделі і т.д.). Найбільш поширеними тестовими артефактами є:

- План тестування (Test Plan) – це документ описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розклади, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

- Набір тест кейсів і тестів (Test Case&Test suite) – це послідовність дій, по якій можна перевірити чи відповідає тестована функція встановленим вимогам.
- Дефекти /Баг Репорт (Bug Reports/Defects) – це документи, що описують ситуацію або послідовність дій призвела до некоректної роботи об'єкта тестування, із зазначенням причин і очікуваного результату.

1.2 Тест кейс

Тест-кейс – це професійна документація тестувальника, послідовність дій, спрямована на перевірку будь-якого функціоналу, що описує як прийти до очікуваного результату. Під тест-кейсом також може матися на увазі відповідний документ, який являє собою формальний запис тест-кейса. Також виділяють:

– високорівневий тест-кейс (high level test case) – тест-кейс без конкретних вхідних даних та очікуваних результатів. Як правило, такий тест-кейс обмежується загальними ідеями та операціями, схожий за своєю суттю з докладно описаними пунктами чекліста. Досить часто зустрічається в інтеграційному і системному тестуванні, а також на рівні димового тестування. Може служити відправною точкою для проведення дослідного тестування або для створення низькорівневих тест-кейсів.

– низькорівневий тест-кейс (low level test case) – тест-кейс з конкретними вхідними даними та очікуваними результатами. Являє собою «повністю готовий до виконання» тест-кейс і є найбільш класичним видом тест-кейсів. Початківців тестувальників найчастіше вчать писати саме такі кейси, тому що прописати всі дані докладно набагато простіше, ніж зрозуміти, якою інформацією можна знехтувати, при цьому не знизивши цінність тест-кейса.

У тестуванні слово «кейс» слід перекладати як «ситуація» просто тому, що тест-кейс – це ситуація, яку потрібно створити для перевірки роботи

однієї окремо взятої функціональності. Всі кроки, перераховані в кейсі, дають можливість перевірити лише одну тестову ситуацію, лише один сценарій використання ПЗ.

Тест-кейс не повинен містити:

1) Залежностей від інших тест-кейсів. Пов'язаний тест-кейс завжди може бути видалений через непотрібність або він може бути змінений, в цьому випадку стане незрозуміло, як виконати тест-кейс, в якому є посилання. Також, через залежність тест-кейсів, може виникнути відчуття, що тестований продукт вже приведений до потрібного стану завдяки виконанню пов'язаних тест-кейсів.
2) Нечітких формулювань. Якщо опис заголовку буде нечітким, то це, як мінімум, ускладнить проходження тест-кейса і сприйняття тест-сюта в цілому.

При створенні теми тест-кейса потрібно пам'ятати:

1) те, що очевидно зараз, може стати абсолютно незрозумілим через декілька місяців. Так, скорочення з нерозшифрованими абревіатурами, зрозумілими зараз, можуть згодом стати китайською грамотою для вас самих, тому простіше буде написати тест-кейс заново, ніж пробиратися через нетрі необачно зроблених скорочень, як в темі, так і всередині кейса;
2) Зайвої деталізації. Надлишкова інформація лише ускладнює розуміння вкладеного в тему сенсу.Тестувальник при виконанні тест-кейсу спочатку прочитає заголовок, усвідомить його, вже приблизно уявить, що йому зараз потрібно буде виконати, і тільки після цього перейде до кроків.

   Атрибути тест-кейсу:

- **ID** (номер тест-кейсу): унікальний ідентифікатор для тест-кейсу, що допомагає управляти і відстежувати його стан.
- **Summary** (назва тест-кейсу): короткий опис мети тесту та функціональності, яка тестується.

- **Description** (опис тест-кейсу): докладний опис сценарію тесту, включаючи передумови, кроки, дії користувача та очікувані результати.
- **Priority** (пріоритет): вказує важливість тест-кейсу в межах тестової набору. Це може бути високий, середній або низький пріоритет.
- **Preconditions** (передумови): умови або стан системи, необхідні для успішного виконання тест-кейсу.
- **Input data** (вхідні дані): параметри або дані, необхідні для виконання тест-кейсу, такі як значення, діапазони або конфігурація системи.
- **Steps** (кроки тесту): послідовність дій, які потрібно виконати для виконання тест-кейсу. Кожен крок має бути чітким та однозначним.
- **Expected result** (очікуваний результат): опис очікуваних результатів або поведінки системи після виконання кожного кроку тесту.
- **Status** (статус): вказує поточний статус тест-кейсу, такий як "Пройдено", "Не пройдено", "У процесі" або "Призупинено".

1.3 Автоматизоване тестування

Автоматизоване тестування передбачає використання інструменту автоматизації для виконання набору тестів. У той час як ручне тестування виконується людиною, що сидить перед комп’ютером, ретельно виконує всі етапи тестування.

Автоматизація програмного забезпечення також може вводити тестові дані в систему, яку тестують, порівнювати очікувані та фактичні результати та генерувати детальні звіти про тестування. Однак воно вимагає значного вкладання коштів та ресурсів.

Цикл розробки вимагає багаторазового виконання одного й того ж набору тестів під час послідовності розробки. Використовуючи автоматизацію, можна написати набір тестів і відтворювати його повторно у разі необхідності. Як тільки набір тестів автоматизовано, втручання людини

не потрібне. Також це допомагає поліпшити ROI (коефіцієнт окупності інвестицій). Метою автоматизації є скорочення кількості тестів, які потрібно запускати вручну, а не усунення ручного тестування в цілому.

Автоматизоване тестування програмного забезпечення є важливим з наступних причин:

- Ручне тестування усіх робочих процесів, усіх полів, усіх негативних сценаріїв вимагає багато часу та грошей.
- Доволі складно протестувати мультимовні сайти вручну.
- Автоматизація не вимагає втручання людини. Ви можете запустити автоматичний тест без нагляду (наприклад вночі).
- Автоматизація збільшує швидкість виконання тесту.
- Автоматизація допомагає збільшити покриття тестами (Test Coverage).
- Ручне тестування може бути нудним а, отже, веде до випадкових помилок.

1.4 Selenium Web Driver

Зазвичай для автоматизації UI тестів використовують Selenium Web Driver. Це колекція API з відкритим кодом, яка використовується для тестування веб-додатків. Інструмент Selenium Webdriver використовується для автоматизації тестування веб-додатків, щоб переконатися, що вони працюють належним чином чи ні. В основному він підтримує такі браузери, як Firefox, Chrome, Safari та Internet Explorer. Це також дозволяє виконувати кросбраузерне тестування.

Web Driver також дозволяє використовувати мову програмування для створення тестових сценаріїв (неможливо в Selenium IDE).

Selenium Webdriver безпосередньо викликає команди браузера, використовуючи рідне для кожного конкретного браузера API. Як відбуваються ці виклики і які функції вони виконують залежить від

конкретного браузера. Запускати скрипти з ним можна як локально, так і віддалено.

Для роботи з Selenium Webdriver необхідно:

1. Браузер — це реальний браузер, роботу якого користувач хоче автоматизувати. Chrome, Firefox, Opera, Safari, IE тощо певної версії, встановленого на певній ОС зі своїми налаштуваннями (за замовчуванням або кастомними). Хоча Webdriver може працювати і з “несправжніми” браузерами, але детально про них розповімо іншого разу, обіцяємо.

2. Driver — найважливіша сутність, запускає браузер і відправляє йому команди, а також закриває його. Насправді є веб сервером. У кожного браузера свій driver. Пов’язано це з тим, що у кожного браузера свої відмінні команди управління і реалізовані вони по-своєму. Як налаштувати, знайти і завантажити список доступних драйверів за посиланням:

3. Скрипт — це тест, який містить набір команд певною мовою програмування для драйвера браузера. Такі скрипти використовують Selenium Webdriver bindings (готові бібліотеки, про які ми щойно згадували), доступні користувачам на різних мовах.

4. Webelement — друга важлива сутність, що представляє собою абстракцію над веб-елементами (кнопки, посилання, інту та ін.).

5. By — абстракція над локатором веб елемента. Цей клас інкапсулює інформацію про селектор (наприклад, CSS), а також сам локатор елемента, тобто усю інформацію, необхідну для знаходження потрібного елемента на сторінці.

Нижче наведено список команд браузера, доступних у C# з Selenium.

Назва команди	Опис	Синтаксис
Url Command	Ця команда використовується для відкриття вказаної URL-адреси в браузері.	`driver.Url = "https://www.guru99.com"`
Title Command	Ця команда використовується для отримання назви веб-сторінки, яка зараз відкрита	`String title = driver.Title`
PageSource Command	Ця команда використовується для отримання вихідного коду веб-сторінки, яка зараз відкрита.	`String pageSource = driver.PageSource`

Close Command	Ця команда використовується для закриття нещодавно відкритого екземпляра браузера.	`driver.Close();`
Quit Command	Ця команда використовується для закриття всіх відкритих екземплярів браузера	`driver.Quit();`
Back Command	Ця команда використовується для переходу до попередньої сторінки історії браузера.	`driver.Navigate().Back();`
Forward Command	Ця команда використовується для переходу до наступної сторінки історії браузера.	`driver.Navigate().Forward()`
Refresh Command	Ця команда використовується для оновлення браузера.	`driver.Navigate().Refresh()`

Webelement представляє всі елементи веб-сторінки. Вони представлені тегами HTML. Кожна кнопка, текстове поле, посилання, зображення, таблиця та фрейм підпадають під Webelements. Операції над веб-елементами можна запускати за допомогою інтерфейсу IWebelement. Щоб взаємодіяти з Webelement, нам потрібно знайти елемент на веб-сторінці, а потім виконати над ним операції. Такі інструменти, як Firebug і Firepath, можна використовувати для визначення Xpath Webelement.

Нижче наведено список команд Webelement, доступних у C#.

Назва команди	**Опис**	**Синтаксис**
Click command	Ця команда використовується для клацання на Webelement. Щоб елемент можна було натиснути, він має бути видимим на веб-сторінці.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); element.Click();`
Clear command	Ця команда спеціально використовується для очищення наявного вмісту текстових полів.Ця команда також використовується для операцій із прапорцями та перемикачами.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); element.Clear();`
SendKeys command	Ця команда використовується для введення значення в текстові поля. Значення, яке потрібно ввести, має бути передано як параметр	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); element.SendKeys("guru99");`

Displayed command	Ця команда використовується для визначення того, чи відображається певний елемент на веб-сторінці. Ця команда повертає логічне значення; true або false залежно від видимості веб-елемента.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); Boolean status = element.Displayed;`
Enabled command	Ця команда використовується, щоб визначити, чи ввімкнено певний веб-елемент на веб-сторінці. Ця команда повертає логічне значення; істинне чи хибне в результаті.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); Boolean status = element.Enabled;`
Selected command	Ця команда використовується, щоб визначити, чи вибрано певний веб-елемент. Ця команда використовується для прапорців, перемикачів і операцій вибору.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); Boolean status = element.Selected;`
Submit command:	Ця команда подібна до команди клацання. Різниця полягає в тому, чи має форма HTML кнопку з типом Надіслати. Тоді як команда click натискає будь-яку кнопку, команда submit натискає лише кнопки з типом submit.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); element.submit();`
Text command	Ця команда повертає внутрішній текст Webelement. У результаті ця команда повертає рядкове значення.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); String text=element.Text;`
TagName command	Ця команда повертає тег HTML веб-елемента. Він повертає рядкове значення як результат.	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); String tagName = element.TagName;`
GetCSSValue Command:	Цей метод використовується для повернення кольору веб-елемента у вигляді рядка rgba (червоний, зелений, синій і альфа).	`IWebelement element = driver.FindElement(By.xpath("xpath of Webelement")); String color = element.getCSSValue;`

Також для роботи з Web Driver використовується фреймворк Nunit. Він є популярним фреймворком тестування для платформи .NET. Він надає зручний спосіб писати, виконувати і керувати тестами одиниць програмного забезпечення (unit tests) в середовищі .NET.

NUnit надає широкі можливості для написання автоматизованих тестів, які дозволяють перевіряти функціональність окремих методів, класів або навіть цілі підсистеми програмного забезпечення. Використовуючи NUnit, розробники можуть створювати тести, які перевіряють правильність

поведінки коду, включаючи очікувані результати, обробку винятків та інші критерії.

Основні можливості NUnit включають:

- Анотації тестів: NUnit надає набір анотацій (атрибутів), які можна застосовувати до методів для позначення їх як тестів одиниць програмного забезпечення.
- Перевірка умов: За допомогою NUnit можна перевіряти, чи виконуються певні умови, використовуючи вбудовані методи перевірки, такі як Assert.AreEqual(), Assert.IsTrue(), Assert.IsFalse() та інші.
- Групування тестів: NUnit дозволяє групувати тести в набори тестів (test fixtures), які відповідають певним класам або компонентам програмного забезпечення. Це дозволяє логічно організувати тести і виконувати їх разом.
- Параметризовані тести: NUnit підтримує параметризовані тести, які дозволяють виконувати однаковий тестовий метод з різними наборами параметрів.
- Пристрої (fixtures): NUnit надає можливість виконувати код перед і після виконання окремих тестів або наборів тестів. Це дозволяє налаштовувати початкові умови для тестів.

2. ПРИКЛАД ЗАВДАННЯ

Частина 1 – Написання тестової документації (тест кейсів)

У цій частині потрібно написати мануальні тест кейси згідно вашого варіанту.

*Тест кейси можна писати українською або англійською мовами.

Частина 2 – Автоматизація тест кейсів

У цій частині вам потрібно автоматизувати ваші мануальні тест кейси **з частини 1** згідно варіанту.

Середовище: .Net, Selenium Web Driver, nUnit,Java, jUnit, JavaScript, MochaJS, Visual Studio.

Завдання виконувати для сайту: https://www.pureformulas.com/

3. ПРИКЛАД РЕАЛІЗАЦІЇ

3.1 Мануальний тест кейс

ID
1
PRECONDITIONS
User has registered account
SUMMARY
Verify that user is able to login into account
TEST STEPS
1. Enter valid login credentials (username and password). *2. Click on the "Login" button.*
EXPECTED RESULT
1. System redirects user to the homepage *2. System shows message 'Hi <user>' message*

3.2 Реалізація на мові програмування C#

Тестовий випадок - Add Product to the Shopping Cart

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System;
using System.Threading;

namespace UnitTest1
{
    [TestFixture]
    public class ManualTests
    {
        private IWebDriver driver;

        [SetUp]
        public void Setup()
        {
            // Ініціалізація веб-драйвера
            driver = new ChromeDriver();
            driver.Navigate().GoToUrl("https://www.pureformulas.com");

            // Виконати вхід в систему

            IWebElement loginSignInButton = driver.FindElement(By.LinkText("Log
In/Sign Up"));
            loginSignInButton.Click();
```

```
            string email = "test@test.com";
            string password = "P@ssword!";

            System.Threading.Thread.Sleep(3000);

            IWebElement emailField = driver.FindElement(By.Id("login_email"));
            emailField.SendKeys(email);

            IWebElement passwordField = driver.FindElement(By.Id("login_password"));
            passwordField.SendKeys(password);

            IWebElement loginButton = driver.FindElement(By.Id("loginButton"));
            loginButton.Click();

             // Перевірка входу
            string currentURL = driver.Url;
            string expectedURL = "https://www.pureformulas.com/account/landing.jsp";
            if (currentURL.Equals(expectedURL, StringComparison.OrdinalIgnoreCase))
            {
                Console.WriteLine("Успішний вхід в систему!");
            }
            else
            {
                Console.WriteLine("Помилка: Вхід не виконано.");
            }

        }

        [TearDown]
        public void TearDown()
        {
            // Вихід з акаунту
            driver.FindElement(By.LinkText("Hi, User")).Click();
            System.Threading.Thread.Sleep(1000);

            IWebElement logoutButton =
driver.FindElement(By.XPath("//li[contains(@class, 'red')]/a[@aria-label='Log
Out']"));
            IJavaScriptExecutor jsExecutor = (IJavaScriptExecutor)driver;
            jsExecutor.ExecuteScript("arguments[0].click();", logoutButton);

            // Перевірка виходу
            string currentURL = driver.Url;
            string greetings =
"https://www.pureformulas.com/account/login.jsp?to=/account/landing.jsp&_requestid=";
            if (currentURL.Contains(greetings))
            {
                Console.WriteLine("Успішний вихід з системи!");
            }
            else
            {
                Console.WriteLine("Помилка: Вихід не виконано.");


            // Завершення роботи з веб-драйвером
            driver.Quit();
            driver.Dispose();
        }

        [Test]
        public void AddProductToShoppingCart()
```

```
        {
            // Крок 1: Перейти на сторінку продукту
            driver.Navigate().GoToUrl("https://www.pureformulas.com/taurox-4x-80-
pellets-by-allergy-research-group.html");

            System.Threading.Thread.Sleep(3000);

            // Крок 2: Додати продукт в корзину
            IWebElement addToCartButton =
driver.FindElement(By.XPath("//button[contains(@aria-label, 'Add to Cart')]"));
            IJavaScriptExecutor jsExecutor = (IJavaScriptExecutor)driver;
            jsExecutor.ExecuteScript("arguments[0].click();", addToCartButton);


            System.Threading.Thread.Sleep(2000);
            driver.FindElement(By.Id("minicart-li")).Click();

            // Почекати на завантаження сторінки корзини
            System.Threading.Thread.Sleep(2000);

            // Крок 3: Перевірити, що продукт додано до корзини

            // Знайти елемент, який містить кількість товарів у кошику
            IWebElement cartCountElement =
driver.FindElement(By.XPath("//span[@class='my-cart-label']"));

            // Отримати текст з елемента, що містить кількість товарів у кошику
            string cartCountText = cartCountElement.Text;

            // Перевірити, чи містить текст значення, що вказує на наявність товару в
кошику
            int cartItemCount;
            bool isItemAdded = int.TryParse(cartCountText, out cartItemCount) &&
cartItemCount > 0;

            // Виконати дії залежно від результату перевірки
            if (isItemAdded)
            {
                Console.WriteLine("Товар був успішно доданий в кошик!");
            }
            else
            {
                Console.WriteLine("Помилка: товар не був доданий в кошик.");
            }

            // Крок 4: Перевірити, що обраний продукт співпадає із доданим до корзини

            // Знайти елемент, який містить інформацію про товар у кошику
            IWebElement cartItemElement =
driver.FindElement(By.XPath("//tr[contains(@class, 'item-item')]"));

            // Знайти елемент, який містить назву товару
            IWebElement itemNameElement =
cartItemElement.FindElement(By.XPath(".//span[@class='item-title']"));

            // Отримати текст з елемента, що містить назву товару
            string itemName = itemNameElement.Text;

            // Перевірити, чи співпадає назва товару з вашим товаром
            string expectedItemName = "Taurox™ 4X - 80 Pellets";

           // Замініть це значення на очікувану назву товару
```

```
            Assert.itemName.Equals(expectedItemName,
StringComparison.OrdinalIgnoreCase);

            // Виконати дії залежно від результату перевірки
            if (isItemMatched)
            {
                Console.WriteLine("Товар співпадає з очікуваним товаром!");
            }
            else
            {
                Console.WriteLine("Помилка: товар не співпадає з очікуваним
товаром.");
            }

        }

    }

}
```

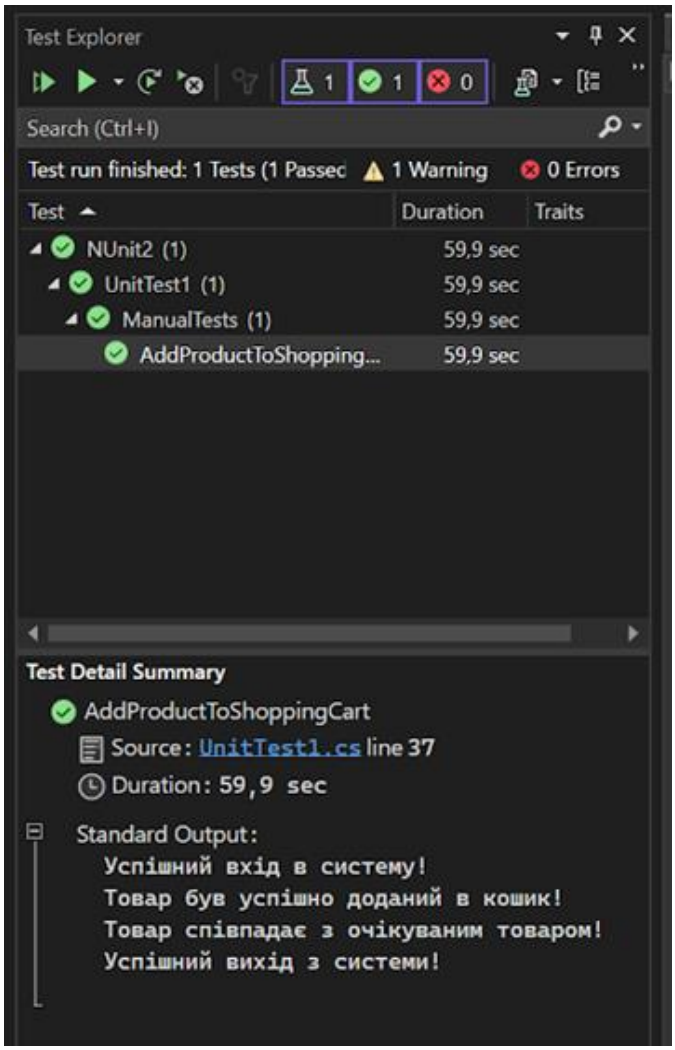

Рисунок 1 – Результат виконання тест кейсу «Add Product to the Shopping Cart»

3.3 Реалізація на мові програмування Java

Тестовий випадок - My Profile - Edit First Name and Save

```
import org.junit.Before;
import org.junit.Test;
import org.junit.After;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class TestingSite {
    private WebDriver driver;

    @Before
    public void setUp() throws InterruptedException {
        System.setProperty("webdriver.chrome.driver",
"G:\\WebDriver\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.pureformulas.com/");
        Thread.sleep(1500);
        WebElement accountLink =
driver.findElement(By.cssSelector("#header > div > div >
div.headerLinks > a.headerLink.accountLink"));
        accountLink.click();
        Thread.sleep(1500);

driver.findElement(By.id("login_email")).sendKeys("testuser@test.com");

driver.findElement(By.id("login_password")).sendKeys("P@ssword!");

        WebElement CookieKlick =
driver.findElement(By.cssSelector("body > div.cc-window.cc-banner.cc-
type-info.cc-theme-classic.cc-bottom.cc-color-override-1898531862 > div
> a"));
        CookieKlick.click();

        driver.findElement(By.className("btn")).click();

    }
    @Test
    public void testEdit() throws InterruptedException {
        WebElement ProductsButton =
driver.findElement(By.cssSelector("#logo > img"));
        ProductsButton.click();

        WebElement CategoriesButton =
driver.findElement(By.cssSelector("#nav > li:nth-child(2) > a"));
        CategoriesButton.click();

        WebElement BloodPressureButton =
driver.findElement(By.cssSelector("#content > div:nth-child(11) >
ul:nth-child(1) > li:nth-child(6) > ul > li:nth-child(2) > a"));
        BloodPressureButton.click();

        WebElement ProductButton =
driver.findElement(By.cssSelector("#custom-featured-category > ul >
li:nth-child(2) > span.item-title > a"));
```

```
        ProductButton.click();

        WebElement addToFavoritesButton =
driver.findElement(By.cssSelector("#addToFavoritesButton"));
        addToFavoritesButton.click();
        WebElement accountLink =
driver.findElement(By.cssSelector("#header > div > div >
div.headerLinks > a.headerLink.accountLink"));
        accountLink.click();

        WebElement FavoritesButton =
driver.findElement(By.cssSelector("#sidebar > ul > li:nth-child(4) >
a"));
        FavoritesButton.click();

    }
    @After
    public void TearDown() throws InterruptedException {
        WebElement LogOut = driver.findElement(By.cssSelector("#sidebar
> ul > li.red > a"));
        LogOut.click();
        Thread.sleep(5000);
        driver.quit();
    }
}
```

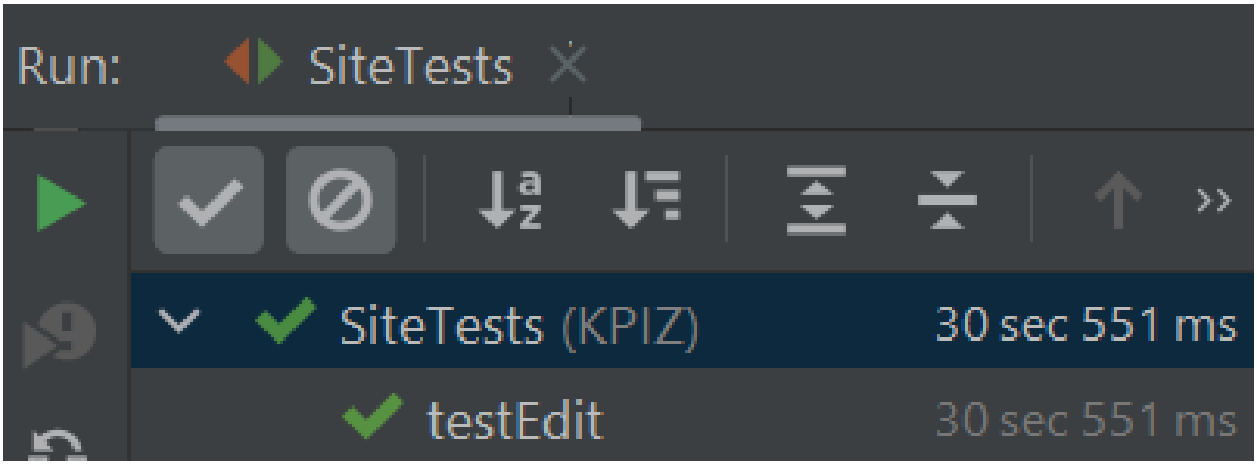

Рисунок 2 – Результат виконання тест кейсу «Edit First Name and Save»

3.4 Реалізація на мові програмування JavaScript

Тестовий випадок - Search for the item

index.spec.js:

```
const{ By, Builder, Browser, until, Key } = require("selenium-webdriver");
const{ suite } = require("selenium-webdriver/testing");
const assert = require("assert");
require("geckodriver");
require("chromedriver");
constlogIn = require("./tests/login");
constlogOut = require("./tests/logout");

constSearchWithoutResults =require("./tests/SearchWithoutResults");
```

```
constSearchWithSubcategories =require("./tests/searchWithSubcategories");
constSearchWithoutSubcategories =require("./tests/SearchWithoutSubcategories");

suite(
	function (env) {
	describe('"PureFormulas" site testing', () => {
	let driver;
	const login = "testuser@test.com";
	const password = "P@ssword!";
	const request = "vitamin";
	beforeEach(async () => {
	driver = await new Builder().forBrowser("chrome").build();
	await logIn(driver, login, password);
	await driver.findElement(By.id("logo")).click();
	await driver.wait(
	until.titleIs("PureFormulas - Health Supplements & Vitamins forSale"),3000
);
});
afterEach(async () => {
	await logOut(driver);
	await driver.quit();
});
it("Search With Subcategories", async () => {
	const value = await SearchWithSubcategories(driver);
	assert.equal("PureFormulas Vitamin A - 100 Softgels",value);
});
	it("Search Without Subcategories", async () => {
	const value = await SearchWithoutSubcategories(driver);

	assert.equal("Thorne Basic Nutrients 2/Day – 60Capsules", value);
});
	it("Search without results", async () => {
	const value = await SearchWithoutResults(driver);
	assert.equal("Can'T Find What You Are Looking For? Try One OfThese...",value);
		});
	});
},
{ browsers: [Browser.CHROME, Browser.FIREFOX] }
);
```

login.js:

```
const{ By, until } = require("selenium-webdriver");
constlogIn = async (driver, login, password) => {
await driver.get("https://www.pureformulas.com/");
await driver.findElement(By.className("cc-btn cc-dismiss")).click();
await driver.findElement(By.className("headerLinkaccountLink")).click();
await driver.wait(until.elementLocated(By.id("loginContent")));
await driver.findElement(By.id("login_email")).sendKeys(login);
awaitdriver.findElement(By.id("login_password")).sendKeys(password);
await driver.findElement(By.id("loginButton")).click();
await driver.wait(until.titleIs("My Account"));
};
module.exports = logIn;
```

logout.js:

```
const{ By, until } = require("selenium-webdriver");
constlogOut = async (driver) => {
await driver.findElement(By.className("headerLinkaccountLink")).click();
await driver.wait(until.elementLocated(By.className("account-title")));
awaitdriver.findElement(By.xpath('//*[@id="sidebar"]/ul/li[12]/a')).click();
};
module.exports = logOut;
```

SearchWithoutSubcategories.js:

```
const { By, until, Key } = require("selenium-webdriver");
const SearchWithoutSubcategories = async (driver) => {
const searchInput = await driver.findElement(By.id("header-
search-term"));
await searchInput.sendKeys("Multivitamin", Key.ENTER);
await driver.wait(until.elementLocated(By.id("content")));
await driver.findElement(By.xpath('//*[@id="custom-featuredategory"]/ ul/li[1]/
span[1]/a')).click();
await driver.wait(until.elementsLocated(By.className("pdp-overview")));

return await driver.findElement(By.xpath('//*[@id="body"]/div/div/div[5]/h1'))
.getText();
};
module.exports = SearchWithoutSubcategories;
```

SearchWithSubcategories.js:

```
const { By, until, Key } = require("selenium-webdriver");
const SearchWithSubcategories = async (driver) => {
const searchInput = await driver.findElement(By.id("header-search-term"));
await searchInput.sendKeys("vitamin", Key.ENTER);
await driver.wait(until.elementLocated(By.id("content")));
await driver.findElement(By.xpath('//*[@id="content"]/div[2]/div/div/div[1]/div/
div[2]/div[3]/p/a')).click();
await driver.wait(until.elementsLocated(By.className("item-item")));
await driver.findElement(By.xpath('//*[@id="custom-featured-category"]/ul/li[1]
/span[1]/a')).click();
await driver.wait(until.elementsLocated(By.id("imagePlus")));
const element = await driver.findElement(By.xpath('//*[@id="body"]/div/div[5]/h1'));
return await element.getText();
};
module.exports = SearchWithSubcategories;
```

SearchWithoutResults.js:

```
const { By, until, Key } = require("selenium-webdriver");
const SearchWithoutResults = async (driver) => {
const searchInput = await driver.findElement(By.id("header-search-term"));

await searchInput.sendKeys("qwerty", Key.ENTER);
await driver.wait(until.elementLocated(By.id("content")));
return await driver.findElement(By.xpath('//*[@id="content"]/h3')).getText();
};
```

```
module.exports = SearchWithoutResults;
```

Рисунок 3 – Результат виконання тест кейсу «Search for the item»

4. ВАРІАНТИ ЗАВДАНЬ

№	Мануальні тест кейси та написати автоматизацію для:
1	● Login ● My Account – Add New Shipping Address ● Logout
2	● Login ● My Account - Create New Wish List ● Logout
3	● Login ● My Account –Add Interests ● Logout
4	● Login ● My Account – Delete Wish List ● Logout
5	● Login ● My Account – Add New Credit Card ● Logout
6	● Login ● My Account – Add Promotional Card Number ● Logout
7	● Login ● My Account – Edit Shipping Address ● Logout
8	● Login ● My Account – Delete Shipping Address ● Logout
9	● Login ● My Account –Add Basic Information ● Logout
10	● Login ● My Account –Change my Password ● Logout
11	● Login ● Search for product ● Logout
12	● Login ● Add Product to the Shopping Cart ● Logout
13	● Login ● Cart – Add to Favorites ● Logout
14	● Login ● Cart – Remove From Cart

	● Logout
15	● Login ● Cart – Increase/Decrease product quantity ● Logout
16	● Login ● Cart – Add Item ● Logout
17	● Login ● Checkout – Edit Address ● Logout
18	● Login ● Checkout– Add a New Address ● Logout
19	● Login ● Checkout – Select Shipping Method ● Logout
20	● Login ● Checkout – Add Comment ● Logout

5. РЕКОМЕНДОВАНІ ДЖЕРЕЛА ІНФОРМАЦІЇ

1 Dorothy Graham, Rex Black, Erik van Veenendaal: “Foundations of Software Testing ISTQB Certification, 4th edition”, Cengage Learning EMEA - 2019

2 Bernd Ruecker: “Practical Process Automation. 1st Ed”, O'Reilly Media - 2021

3 Arnon Axelrod: “Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects 1st ed.”, Print2print - 2018

4 Narayanan Palani: “Advanced Selenium Web Accessibility Testing: Software Automation Testing Secrets Revealed”, Momentum Press - 2019

5 Онлайн ресурс -https://www.guru99.com/automation-testing.html

6 Онлайн ресурс - https://www.softwaretestinghelp.com/automation-testing-tutorial-1/

7 Онлайн ресурс - https://www.selenium.dev/documentation/webdriver/

8 Онлайн ресурс - https://docs.nunit.org/

9 Онлайн ресурс - https://junit.org/junit5/docs/current/user-guide/

10 Онлайн ресурс - https://mochajs.org/

Навчально-методичне видання

Тимчишин В.С., Крепич С.Я., Співак І.Я.,
Мельник А.М., Манжула В.І.,

МЕТОДИЧНІ ВКАЗІВКИ

до виконання
комплексного практичного індивідуального завдання
з дисципліни

«ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ»

здобувачів ступеня вищої освіти «бакалавр»
спеціальностей 121 «Інженерія програмного забезпечення»
126 «Інформаційні системи та технології»

Підписано до друку 20.06.2023 р.
Формат 60х84/16. Папір офсетний.
Друк офсетний. Зам. № 23-625
Умов.-друк. арк. 1,1. Обл.-вид. арк. 1,3.
Тираж 30 прим.

Віддруковано ФО-П Шпак В. Б.
Свідоцтво про державну реєстрацію В02 № 924434 від 11.12.2006 р.
м. Тернопіль, бульвар Просвіти, 6/4. тел. 097 299 38 99.
E-mail: tooums@ukr.net

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 7599 від 10.02.2022 р.