Харківський національний університет імені В.Н. Каразіна

Факультет комп'ютерних наук

Безпека інформаційних систем і технологій

«Допущено до захисту»

Зав.кафедрою БІСТ

Сватовський І.І._____

« » червня 2023р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

спеціальність: 125 Кібербезпека

на тему: «Система автоматизованого тестування web-додатків»

оцінка « » Керівник доц. Мелкозьорова О.М.

(прізвище та ініціали/підпис)

Голова ЕК Рецензент д.т.н. Краснобаєв В.А.

(прізвище та ініціали/підпис)

Лемешко О.В._____ Виконавець студент групи КБ-42

Пунтус Є.В.

(прізвище та ініціали/підпис)

Харків – 2023

РЕФЕРАТ

Дипломна робота присвячена розробці системі автоматизованого тестування веб-додатків. Робота складається зі вступу, трьох розділів, висновків до кожного розділу, переліку посилань та додатка. Пояснювальна записка містить: 54 сторінок, 10 рисунків, 2 таблиці, 1 додаток, 24 джерел.

Мета дипломного проекту – дослідження теорії автоматизованого тестування веб-додатків та створення системи з використанням автотестів.

Завдання дипломного проекту – створити систему автоматизованого тестування web-додатка. Проаналізувати документацію, розробити тест-кейси. Написати скрипт на мові програмування C# в інструменті Selenium.

Об’єкт проектування – система авто тестів Web-додатків.

Програмне забезпечення для виконання завдання: Selenium Webdriver, Microsoft Visual Studio 2022.

Результати дипломного проектування рекомендується використовувати при розробці нових систем автотестів, які надають надійність перевірки функціональності Web-додатків.

Ключові слова: АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ, АВТОТЕСТИ, ТЕСТ-ПЛАН, SELENIUM, C#.

ABSTRACT

The diploma thesis is dedicated to the development of an automated testing system for web applications. It consists of an introduction, three chapters, conclusions for each chapter, a list of references, and appendices. The explanatory note contains: 54 pages, 10 figures, 2 tables, 1 appendix, 24 sources.

The aim of the diploma project is to study the theory of automated testing of web applications and create a system using automated tests.

The tasks of the diploma project are to create an automated testing system for a web application, analyze documentation, develop test cases, and write a script in the C# programming language using the Selenium tool.

The object of the design project is the automated testing system for web applications.

Software for completing the tasks: Selenium Webdriver, Microsoft Visual Studio 2022.

The results of the diploma project are recommended to be used in the development of new automated testing systems that provide reliable verification of the functionality of web applications.

Keywords: AUTOMATED TESTING, AUTOMATED TESTS, SELENIUM, TEST-PLAN, C#.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ПК	-	Персональний комп’ютер
ПЗ	-	програмне забезпечення
IDE	-	Integrated Development Environment – інтегроване середовище розробки
ОС	-	Операційна система
HTTP	-	Hypertext Transfer Protocol – протокол передачі гіпертексту
HTTPS	-	HyperText Transfer Protocol Secure – захищений протокол передачі гіпертексту
API	-	Application Programming Interface – прикладний програмний інтерфейс
GUI	-	Graphical User Interface – графічний інтерфейс користувача
XML	-	Extensible Markup Languege – розширювана мова розмітки
HTML	-	HyperText Markup Language - мова гіпертекстової розмітки
CSS	-	Cascading Style Sheets - каскадні таблиці стилів

ВСТУП

В даний час розвиток веб-додатків є однією з найбільш перспективних галузей інформаційної технології. Значна кількість програмних продуктів базується на веб-технологіях, що зумовлює високі вимоги до якості та надійності роботи цих додатків. Однак, при швидкому темпі розвитку цієї галузі, збільшується кількість проблем та недоліків, які можуть виникати при їх використанні.

Один із найбільш ефективних способів забезпечення якості веб-додатків - автоматизоване тестування. На сьогоднішній день існує велика кількість інструментів для автоматизованого тестування веб-додатків, однак не всі вони є ефективними та зручними у використанні.

Отже, головною метою даної дипломної роботи є розробка системи автоматизованого тестування веб-додатків, яка буде забезпечувати ефективне та зручне тестування веб-додатків. Галузь застосування результатів даної роботи полягає в покращенні якості веб-додатків та зменшенні кількості помилок під час їх розробки та експлуатації.

Враховуючи зазначені фактори, дана дипломна робота є актуальною та важливою для розробників веб-додатків та тестувальників. Результати роботи допоможуть покращити якість та надійність веб-додатків, знизити витрати на їх розробку та підтримку, а також підвищити ефективність процесу тестування.

Результати даної роботи можуть бути використані для подальшої розробки та вдосконалення інших інструментів автоматизованого тестування веб-додатків.

Для досягнення даної мети в дипломному проекті поставлені та вирішені наступні завдання:

1) Проаналізувати теорії тестування, їх види.
2) Описати переваги та недоліки автоматизованого тестування.
3) Порівняти інструменти для автоматизації.
4) Підключити усі необхідні інструменти.
5) Оформити документацію web-додатку.
6) Створити систему авто тестів.
7) Провести аналіз результатів.

1 СУЧАСНІ ПІДХОДИ ТА МЕТОДИ ТЕСТУВАННЯ ПЗ

1.1 Поняття тестування програмного забезпечення.

Тестування програмного забезпечення - процес експлуатації системи або компонент за певних умов, спостереження або запис результатів, а також проведення оцінки якогось аспекту систему або компонент. Або іншими словами - процес аналізу елемента програмного забезпечення для виявлення відмінностей між існуючими та необхідними умовами (тобто помилками) і оцінки особливості програмних елементів. [1]

У вузькому розумінні тестування – це процес визначення відповідності об’єкта тестування заданим специфікаціям (характеристикам, функціям), який полягає в опрацюванні програмою послідовності різноманітних контрольних наборів тестів з відомими результатами. Тести підбираються так, щоб вони охопили найрізноманітніші типи можливих ситуацій. У контексті розробки, відповідно, тестування розуміється як процес визначення відповідності продукту початковим специфікаціям, які були задані технічним завданням. [2]

Якщо розглядати тестування в широкому сенсі, то його можна характеризувати як процес експериментального аналізу функціональності деякої досліджуваної системи. При цьому постулюється, що всякий випадок тестування можна визначити як дослідження. Наприклад, тестування з метою виявлення структурних дефектів (дефектів конструкції) можна визначити, як дослідження системи на предмет наявності дефектів конструкції. [2]

Будь-яке тестування має на увазі дві діючі особи: суб’єкт і об’єкт тестування. Суб’єктом тестування (тобто джерелом активності) виступає «тестувальник» (tester) – спеціально призначена особа, в обов’язки якої входить виконання тестування. Під об’єктом тестування розуміється

досліджувана система. При цьому в ролі тестованої системи може виступати як весь продукт, так і окремі його модулі, складові частини. [2]

При тестуванні програмного забезпечення, в загальному випадку, об’єктом тестування є версія розроблюваного програмного продукту. [2]

Зрозуміло, що під версією мається на увазі не тільки «фінальна» (кінцева) версія, а й проміжні версії, одержувані в ході розробки. При проведенні тестування завжди має місце «зовнішній вплив» на систему з боку тестувальника, який також називають «тестовим впливом». [2]

Тестування полягає у здійсненні тестувальником спрямованого впливу на систему, що тестується, який передбачає отримання деякої очікуваної реакції, яку називають «тестовою реакцією», поява якої має підтвердити відповідність системи конкретної окремо взятої специфікації. Отримання зворотної (стосовно очікуваної) або неспецифічної реакції дозволяє говорити про невідповідність системи специфікації і приймати рішення про повернення програмного продукту на доопрацювання з метою усунення дефекту. [2]

Як і в моделюванні, у тестуванні поняття системи є ключовим, оскільки наше уявлення про досліджуваний об’єкт є первинним стосовно експерименту. Іншими словами характер експерименту залежить від наших знань про структурний склад і характер зв’язків між компонентами тестованої системи.[2]

Необхідно усвідомлювати, що процес тестування програмного забезпечення включає не тільки виконання тестів, але й багато інших дій, пов'язаних з забезпеченням якості:

- аналіз та планування;
- розробку тестових сценаріїв;
- оцінку критеріїв закінчення тестування;
- написання звітів;

- рецензія документації;
- проведення статичного аналізу.

1.2 Основні типи тестування

Мануальне тестування (manual testing) – це різновид тестування програмного забезпечення, у якому тестувальник програмного забезпечення розробляє та виконує тестові приклади без використання будь-яких інструментів автоматизованого тестування. Основною метою ручного тестування є виявлення проблем, помилок і дефектів програмного забезпечення. Будь-яке нове програмне забезпечення слід протестувати вручну перед виконанням автоматизованого тестування. [3]

Переваги ручного тестування:

- Ручне тестування виявляє майже кожну помилку та проблему програмного забезпечення.
- За допомогою ручного тестування тестувальники програмного забезпечення можуть отримати доступ до візуальних компонентів, таких як макет, текст та інші компоненти. Вони також можуть визначити проблеми UX та UI.
- Це підходить, якщо ми вносимо деякі незаплановані зміни до програм, оскільки їх можна легко прийняти.
- Оскільки ми не використовуємо навичок високого рівня чи тип інструментів, це матиме низьку вартість операцій. [3]

Недоліки ручного тестування:

- Основний недолік ручного тестування полягає в тому, що воно займає багато часу.
- За допомогою ручного тестування непросто виявити комбінацію кольорів і різницю розмірів об’єктів GUI.
- Тестування продуктивності та тестування навантаження неможливі в ручному тестуванні.

- У ручному тестуванні регресійні тести займають багато часу. [3]

Роль тестувальника полягає у відтворенні ролі користувача програми та використанні функцій програми з метою виявлення помилок у її роботі. Для ефективного проведення професійного тестування тестувальник зазвичай користується написаним планом тестування з варіантами тестування (test cases). [3]

Нижче наведено типи ручного тестування:

1) Тестування білого ящика (white box testing):

Також відоме як випробування прозорої коробки, структурної та скляної коробки. У цьому типі тестування розробники використовуватимуть внутрішню перспективу системи та навички програмування для розробки тестів. [3]

2) Тестування чорного ящика (black box testing):

Під час тестування «чорної скриньки» тестувальники оцінюють функціональність програми без урахування внутрішньої структури коду. Тестування «чорного ящика» можна застосовувати до всіх рівнів тестування, таких як інтеграційне, модульне, приймальне та системне тестування. [3]

3) Приймальне тестування (acceptance testing):

Приймальне тестування також відомі як перед серійні випробування. У тестування також приймають участь кінцеві користувачі для перевірки функціональності програми. [3]

4) Модульне тестування (unit testing):

Модульне тестування також відоме як тестування компонентів або тестування модулів. Виконується, щоб перевірити, чи окремий модуль або одиниця коду працює належним чином. [3]

5) Тестування системи (system testing):

Тестування системи – це процес тестування повністю інтегрованої програми для оцінки комфортності системи з її визначеними вимогами, відоме як тестування системи або наскрізне тестування. Перевіряє всю систему, щоб переконатися, що програма працює за планом. [3]

6) Інтеграційне тестування (integration testing):

Інтеграційне тестування — це механізм тестування програмного забезпечення між двома програмними модулями. Виконується за допомогою різних підходів, які називаються «зверху вниз», «знизу вгору» та «великого вибуху». [3]

Для професійного тестування використовується тест-плани з варіантами тестування.

Тест-план (Test plan) - це документ, який містить опис повного обсягу робіт з тестування, починаючи з опису об'єкта тестування, стратегії, розкладу, критеріїв початку та закінчення тестування, до необхідного обладнання та спеціальних знань, які необхідні в процесі роботи. Він також включає оцінку ризиків з варіантами їх вирішення. [4]

Тест-план призначений для:

- врегулювання процесів тестування;
- пріоритезації завдань;
- планування ресурсів;
- обліку програмного забезпечення та людських ресурсів.

Як правило, тест-план складається QA-лідом команди тестувальників. Також, варто зазначити, що тест-план може бути багаторазово відредагований тестувальниками. [4]

Автоматизоване тестування (automated testing) - це застосування програмних інструментів для автоматизації ручного процесу перевірки та підтвердження програмного продукту. Більшість сучасних проектів з

використанням методологій Agile та DevOps включають автоматизоване тестування з самого початку. [5]

У автоматизованому тестуванні використовується три рівні:

1) Тестування на рівні коду (модульне тестування).

2) Функціональне тестування.

3) GUI тестування (Graphical user interface – Графічний інтерфейс користувача).

Для ефективного тестування веб-додатків на професійному рівні використовуються тест-кейси, які забезпечують часткове або повне покриття інструментального середовища. Однак розробка тест-кейсів, запуск тестів, оцінка їх виконання та опис дефектів в баг-трекінгових системах потребують активної участі тестувальника і не можуть бути повністю автоматизовані. [6]

Тест-кейс - це набір дій, які виконуються в системі для визначення, чи задовольняє вона вимоги до програмного забезпечення і працює правильно. Метою тест-кейса є визначення того, чи працюють різні функції в системі так, як очікується, і підтвердження того, що система задовольняє всім відповідним стандартам, правилам та вимогам замовника. Процес написання тест-кейса також може допомогти виявити помилки або дефекти в системі. [6]

Тест кейси визначають, що потрібно зробити для тестування системи, включаючи кроки, що виконуються в системі, значення вхідних даних, що вводяться в систему, та очікувані результати протягом виконання тестового випадку. Використання тестових випадків дозволяє розробникам та тестувальникам виявляти помилки, які можуть виникнути під час розробки або дефекти, які були пропущені під час спонтанних тестів. [6]

Переваги такого виду тестування включають:

- Гарантоване охоплення тестуванням всіх аспектів системи.

- Зменшення витрат на підтримку та супровід програмного забезпечення.
- Можливість повторного використання тестових випадків.
- Підтвердження, що програмне забезпечення задовольняє вимоги кінцевих користувачів.
- Покращення якості програмного забезпечення та досвіду користувачів.
- Висока якість продуктів призводить до більш задоволених клієнтів.
- Більш задоволені клієнти збільшують прибуток компанії.

В цілому, написання та використання тест кейсів призводить до оптимізації бізнесу. Клієнти більш задоволені, збільшується утримання клієнтів, зменшуються витрати на обслуговування та виправлення продуктів, а також виробляються надійніші продукти, що покращує репутацію і образ бренду компанії. [6]

1.3 Види і направлення тестування

Функціональне тестування - це тип тестування програмного забезпечення, яке перевіряє відповідність програмної системи функціональним вимогам/специфікаціям. Метою функціональних тестів є перевірка кожної функції програмного застосунку, шляхом надання відповідних вхідних даних і перевірки вихідних результатів з вимогами щодо функціональності. [7]

Функціональне тестування в основному базується на "чорній скриньці" і не стосується вихідного коду програми. Це тестування перевіряє інтерфейс, API, базу даних, безпеку, комунікацію клієнт/сервер та інші функціональні можливості тестованого програмного застосунку. Тестування може проводитись як вручну, так і за допомогою автоматизації. [7]

Стресове тестування (Stress testing) піддає програму важким навантаженням. Не слід плутати з тестуванням обсягу. Великий стрес - це пік обсягу даних або активності, який зустрічається протягом короткого проміжку часу. Можна провести аналогію з оцінкою письменника: тест обсягу визначить, чи впорається письменник з переробкою великого звіту, а стрес-тест визначить, чи може письменник писати зі швидкістю 50 слів на хвилину. [8]

Оскільки стрес-тестування передбачає елемент часу, воно не застосовне до багатьох програм, наприклад, компілятора або програми обробки пакетних зарплат. Однак, воно застосовне до програм, які працюють з різними навантаженнями, інтерактивними, реального часу та програмами керування процесами. Якщо система повітряного контролю має відстежувати до 200 літаків у своєму секторі, ви можете провести стрес-тест, симулюючи присутність 200 літаків. Оскільки ніщо фізично не перешкоджає 201-му літаку потрапити в сектор, додатковий стрес-тест дослідить реакцію системи на цей неочікуваний літак. Додатковий стрес-тест може симулювати одночасний вхід великої кількості літаків в сектор. [8]

Навантажувальне тестування (Load testing) є найпростішою формою тестування продуктивності. Зазвичай його проводять для оцінки поведінки програми під заданим очікуваним навантаженням, яке може бути, наприклад, очікуваною кількістю одночасних користувачів додатку або зазначеною кількістю транзакцій протягом певного інтервалу часу. Цей тип тестування зазвичай дає змогу отримати час відгуку на всі основні бізнес-транзакції. При спостереженні за базою даних, сервером додатків, мережею тощо, таке тестування може також ідентифікувати обмеження продуктивності. [9].

Тестування зручності (usability testing) - передбачає оцінку продукту або сервісу шляхом залучення представників цільової аудиторії до його тестування. У типовому тестуванні зручності використання учасникам надаються конкретні завдання для виконання, під час яких спостерігачі

спостерігають, слухають та роблять записи. Метою є виявлення будь-яких проблем зручності використання, збір якісних та кількісних даних та оцінка задоволення учасників продуктом. [10]

Тестування зручності використання дозволяє командам дизайну та розробки виявляти проблеми до їх закодування. Чим раніше виявляються проблеми і виправляються, тим менше коштуватимуть виправлення як у відношенні часу співробітників, так і можливого впливу на графік робіт. Під час тестування зручності використання можна:

- Дізнатися, чи можуть учасники успішно виконати визначені завдання.
- Визначити, який час потрібно для виконання визначених завдань.
- Дізнатися, наскільки задоволені учасники вашим веб-сайтом або іншим продуктом.
- Виявити зміни, які потрібно винести для поліпшення продуктивності та задоволеності користувачів.
- Аналізувати продуктивність, щоб переконатися, що вона відповідає вашим цілям зручності використання. [10]

Тестування інтерфейсу користувача або UI-тестування - це діяльність, спрямована на перевірку якості інтерфейсу користувача, а також його відповідності всім нормам і вимогам. Таке тестування може проводитися вручну, а може задіяти спеціальні інструменти, все залежить від цілей та особливостей UI. Тестування UI проводиться паралельно з перевіркою UX програмного продукту, тому вдається досягти більшої міри перевірки та ще більш високих результатів. [11]

Тестування безпеки забезпечує захист програмного застосунку від зловмисних дій та зберігає його функціональність відповідно до задуманого. Воно допомагає програмам переконатися, що їхні чутливі дані/інформація не піддаються порушенню. [12]

Якщо програмний застосунок не є безпечним, і хакер виявляє уразливість у програмі, вона буде використана, що призводить до наступних передбачуваних наслідків:

- Пошкодження репутації організації.
- Негативний вплив на враження клієнтів з можливим ризиком втрати взаємовідносин.
- Додаткові витрати, пов'язані з усуненням уразливості після релізу програмного застосунку. [12]

Шість принципів тестування безпеки:

1) Конфіденційність: еквівалент приватності і включає набір правил, які обмежують доступ до інформації. Вона захищає від розкриття інформації непередбаченим отримувачам і призначена для запобігання потраплянню чутливої інформації до неправильних осіб. Вона гарантує, що інформацію отримує тільки призначена особа, а доступ до неї обмежується для авторизованих користувачів.
2) Цілісність: забезпечення послідовності, точності та надійності даних протягом всього їхнього життєвого циклу і дозволяє передавати точну та бажану інформацію від відправників до призначених отримувачів. Вона гарантує, що дані не можуть бути змінені неавторизованими особами (наприклад, під час порушення конфіденційності).
3) Аутентифікація: підтвердження ідентичності користувача і дозволяє користувачеві мати впевненість, що інформація, яку він отримує, походить з певних відомих джерел.
4) Авторизація: встановлює права доступу для користувачів на основі їхньої ролі.
5) Доступність: готовність інформації за потребою. Іншими словами, інформація повинна бути доступна авторизованим особам, коли вони потребують її. Доступність найкраще забезпечується ретельним обслуговуванням усього обладнання, негайним виконанням ремонтів

обладнання та підтримкою правильно функціонуючого середовища операційної системи без конфліктів програмного забезпечення.

6) Недійсність: забезпечує відсутність заперечень від відправника або отримувача щодо відправлених/отриманих повідомлень. Вона обмінюється інформацією аутентифікації з підтвердженим часовим штампом, наприклад, «ідентифікатор сесії» та інше. [12]

Конфіденційність, цілісність і доступність, також відомі як тріада KID (Конфіденційність, Цілісність, Доступність), є моделлю, призначеною для керування політиками інформаційної безпеки в компанії. Іноді цю модель також називають тріадою AIK, щоб уникнути сплутування з Центральним Розвідувальним Агентством. [12]

Тестування локалізації (Localization testing) - є технікою тестування програмного забезпечення, в якій перевіряється поведінка програмного забезпечення для конкретного регіону, мови або культури. Метою локалізаційного тестування є перевірка відповідності програмного забезпечення відповідним лінгвістичним і культурним аспектам для певного регіону. Це процес налаштування програмного забезпечення відповідно до обраної мови та країни. [13]

Основні області, що піддаються впливу локалізаційного тестування, включають вміст і користувацький інтерфейс. [13]

Це процес тестування глобалізованого додатка, чий інтерфейс користувача, мова за замовчуванням, валюта, формат дати, часу та документація розроблені відповідно до цільової країни або регіону. Це забезпечує можливість використання додатка в даній країні. [13]

Приклади:

1) Якщо проект розроблений для штату Таміл-Наду в Індії, проект повинен бути написаний мовою таміл, має бути присутня віртуальна клавіатура для тамільської мови тощо.

2) Якщо проект розроблений для США, то формат часу повинен відповідати стандартному часу США, а також мова та формат грошей повинні відповідати стандартам США.[13]

1.4 Автоматизація тестування

Автоматизоване тестування, на відміну від ручного, спрощує процес виявлення багів за допомогою спеціальних програм, чим скорочує витрати й час на цикл тестування. [14]

Іншими словами, це дозволяє отримати готовий програмний продукт без багів в коротші терміни, ніж при ручному тестуванні. [14]

Основними цілями впровадження автоматизації в процес тестування є підвищення ефективності, розширення покриття та прискорення тестування шляхом постійного виконання тестових сценаріїв. Автоматичні тести можна запускати регулярно, як у робочий, так і у неробочий час. У середньому, виконання ручних тестів, виявлення та реєстрація помилок займає близько одного дня для тестувальника. Завдяки автоматизації цей процес скорочується до хвилин, а також дозволяє виявляти помилки в коді вже на етапі його внесення до репозиторію вихідного коду. [14]

Процес автоматизації тестів проходить наступним чином:

Перший етап полягає у запуску автоматичних тестів, використовуючи перевірені ручні сценарії. Для автоматизації процесу потрібно вибрати інструмент тестування, враховуючи тип продукту (веб-сайт, мобільний додаток, API, ПЗ і т.д.). При виборі інструменту звертається увага на відповідність бюджету, технічні характеристики та підтримку використовуваних технологій, кваліфікацію фахівців, які працюють з інструментом, і якість звітів, що генеруються. [14]

Другий етап. Передбачає визначення обсягу автоматизації, залежно від стратегії продукту. Для основного продукту рекомендується максимальне

покриття автоматичними тестами, а для прототипу терміни виконання грають важливу роль, а не якість продукту. [14]

Третій етап. Включає розробку стратегії та плану автоматизації. На цьому етапі проектується інфраструктура для автоматизації, готуються необхідні стенди, затверджується графік запуску сценаріїв. Перед запуском автоматичних тестів підготовлюються тестові дані. Після виконання тестів, аналізу звіту та виявлення помилок проводиться відстеження, виправлення та повторне тестування. [14]

Четвертий етап. Передбачає підтримку тестів для перевірки якості автоматизації. Це необхідно для поліпшення ефективності наявних сценаріїв та розробки нових. [14]

Автоматизоване тестування програмного забезпечення є важливим з наступних причин:

- Ручне тестування усіх робочих процесів, полів та негативних сценаріїв вимагає багато часу та грошей.
- Тестувати багатомовні сайти вручну досить складно.
- Автоматизація не потребує втручання людини. Запустити автоматичний тест можна без нагляду.
- Автоматизація збільшує швидкість виконання тесту.
- Автоматизація збільшує покриття тестами.
- Ручне тестування може бути виснажливим і, отже, призводить до випадкових помилок.[15]

Але варто розуміти, що далеко не всі проекти потребують повної автоматизації. Деяким компаніям ефективніше та економічно вигідніше працювати з «ручним» тестуванням, прискоривши його написанням додаткових скриптів. Автоматизація в багатьох проектах поєднується з налагодженою роботою QA фахівців.

1.5 Тестування Web-додатків

Веб-додаток (web application) - це програмне забезпечення, яке запускається на веб-сервері і доступне користувачам через браузер або інші клієнтські програми, що підтримують протокол HTTP. Веб-додаток може зберігати та оброблювати дані на сервері або взаємодіяти з базами даних, відображати статичний або динамічний контент, виконувати складні бізнес-операції, взаємодіяти з іншими веб-додатками, API та іншими системами. [16]

Зовнішній інтерфейс веб програми розробляється за допомогою таких мов програмування: HTML, CSS, Javascript, які підтримуються на будь-якому браузері (Chrome, Opera, Yandex, Mozilla). Для написання серверної частини (Back-end) може використовуватися будь-яка інша мова програмування або фреймворк. Наприклад: Python, PhP, Ruby, Java. [16]

Основні переваги веб-застосунків:

- веб додатки можуть застосовуватися на всіх операційних системах (Linux, Mac, Windows), тому що вони підтримують сучасні браузери;
- їх набагато легше підтримувати, оскільки у веб-додатках використовується той самий код порівняно з desktop додатками;
- так як він не включає багато роботи з елементами ПК (ядро, процесор, відеокарта), веб додаток простіше програмувати;
- на відміну від мобільних додатків, для веб-додатків не потрібно схвалення жодних платформ, щоб випустити свою програму;
- веб додатки це більш економний варіант для будь-якого підприємства;
- оскільки веб-програми не вимагають підписки або покупки ліцензій, а можуть використовуватися як SaaS-сервіс, що значно дешевше. [16]

Етапи тестування веб-проектів:

1) Підготовчий етап та вивчення документації.

В даний етап входить аналіз технічного завдання; вивчення кінцевих макетів; тест кейсів; матриці відповідності (для валідації покриття вимог щодо продукту тестами) і складання плану тестування. [17]

2) Тестування верстки.

Візуальна частина:

- неправильне відображення складових інтерфейсу, блоків, невідповідність колірної гами;
- тестування локалізації (переклад сайту);
- відповідність макету;
- підсвічування полів з помилками;
- перевірка у дозволах (+ прокрутка);

Спосіб перевірки: FirefoxMenu -> Інструменти -> Веб-розробник -> Адаптивний дизайн або Resolution Test Plugin у Chrome. [17]

3) Доступність і відсутність JS помилок.

- Чи натискаються клікабельні елементи (внутрішні/зовнішні посилання, посилання на електронну пошту, кнопки, іконки);
- курсор змінюється тільки під час наведення на клікабельні елементи веб-сторінки, в іншому випадку – ні;
- коректні підказки;
- повинні бути підписи невеликим сірим кольором під час відключення зображень (Web Developer -> Images -> Replace Image With Alt Attributes);
- коректність роботи веб-додатка при відключенні JS скриптів. Критичні функції повинні бути доступні без JS (в Web Developer -> Disable -> Disable JS -> All JS). [17]

Коректна робота, надійна верстка [17]:

- перевірка роботи з даними (введення великої та малої кількості тексту у форму; блоки з контентом міняються місцями (Firebug (HTML -> Edit)));
- перевірка роботи стилів (введення тексту з заголовками, з абзацом і без, з картинками).

404-і запити:

Відсутність помилок типу 404 (Firefox -> Tools -> Validate links). [17]

4) Функціональне тестування.

Вид тестування, у якому виявляється некоректна/неправильна робота функціоналу програми. [17]

Необхідними перевірками є:

- коректність роботи головних функцій сайту;
- перехід за посиланнями;
- перевірка користувальницьких форм (валідація полів, обов’язкові/необов’язкові поля, повідомлення про помилки під час неправильного введення, додавання коментарів у блог, зворотний зв’язок);
- пошук і покупка товару, оформлення замовлення;
- звірка переданого замовником контенту з наявним на сайті;
- перевірка можливої авторизації/реєстрації;
- додавання, видалення та редагування даних користувачів, товарів і замовлень. [17]

Ad-hock тестування – імпровізаційне тестування без підготовки. [17]

Допомагає зрозуміти:

- чи зрозумілим є призначення форм;

- чи відзначені обов’язкові поля та чи всі обов’язкові поля відмічені;
- чи вбудована обов’язкова перевірка заповнених форм;
- чи відбувається перевірка правильності введення контактних даних. [17]

З переваг даного тестування можна виділити:

- достатньо швидке знайомство з системою;
- специфічні несправності;
- масу питань і пропозицій;
- економію часу.[17]

1.6 Архітектура «Клієнт - сервер»

Архітектура клієнт-серверного додатку (рис. 1.2) - це розподілена архітектура, яка складається з двох основних компонентів: клієнтської частини, яка є інтерфейсом користувача, і серверної частини, яка забезпечує зберігання та обробку даних та надає клієнтській частині доступ до цих даних.

Клієнтська частина зазвичай є програмним додатком або веб-додатком, який взаємодіє з сервером через мережу Інтернет або локальну мережу. Серверна частина може бути фізичним комп'ютером або набором комп'ютерів, що забезпечують зберігання та обробку даних.

Архітектура клієнт-серверного додатку забезпечує розподілену обробку даних, що дозволяє зменшити навантаження на окремі компоненти системи та забезпечити більшу масштабованість та доступність для користувачів.

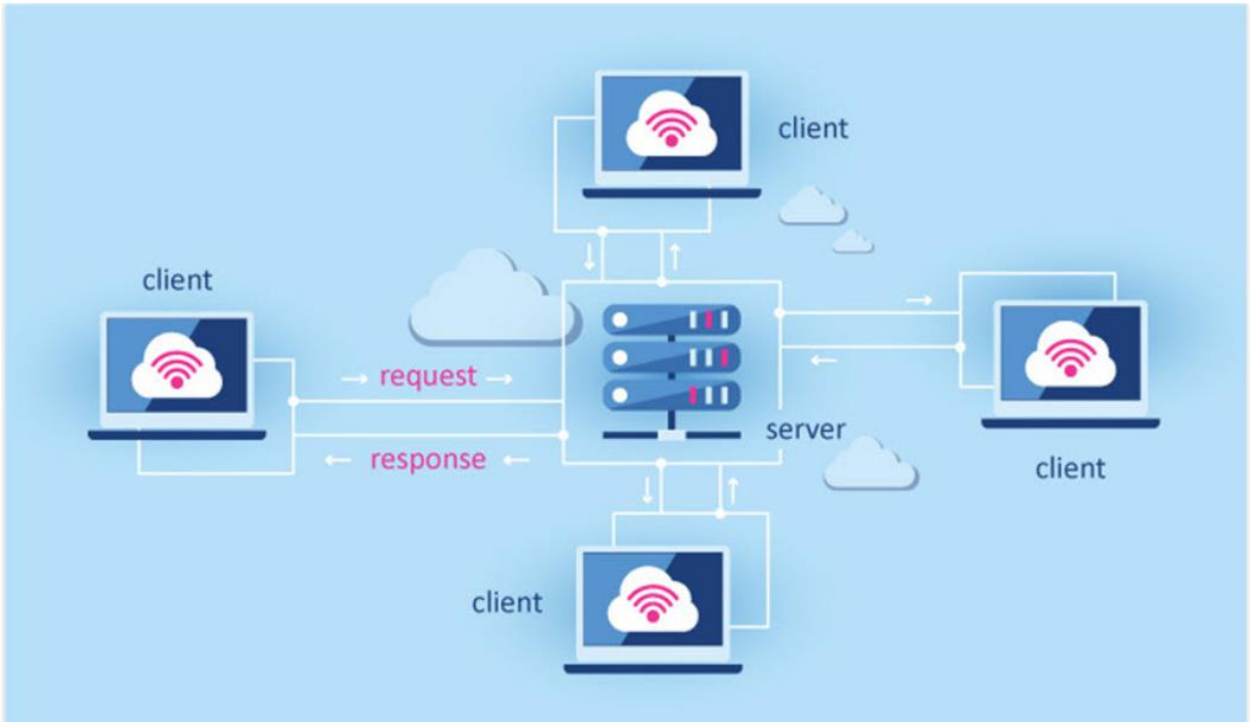

Рисунок 1.2 – Архітектура клієнт-серверного додатку [18]

Модель такої системи полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше. [18]

Функції, які реалізуються на сервері:

- зберігання, доступ, захист і резервне копіювання даних;
- обробка клієнтського запиту;
- відправлення результату (відповіді) клієнту.

Функції, які реалізуються на стороні клієнта:

- надання користувальницького інтерфейсу;
- формулювання запиту до сервера і його відправка;
- отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних). [18]

Клієнт посилає запит на сервер, а він у свою чергу реагує на request і повертає Http response.

HTTP-повідомлення складаються з текстової інформації в кодуванні ASCII, записаної в кілька рядків. У HTTP/1.1 і попередніх версіях вони пересилалися як звичайний текст. У HTTP/2 текстове повідомлення поділяється на кадри, що дозволяє виконати оптимізацію та підвищити продуктивність. [19]

HTTP запити та відповіді мають близьку структуру. Вони складаються з:

- Стартовий рядок, що описує запит, або статус (успіх або збій). Це завжди один рядок.
- Довільного набору заголовків HTTP, що визначають запит або описують тіло повідомлення.
- Порожній рядок, що вказує, що вся мета інформації надіслано.
- Довільного тіла, що містить дані, що пересилаються із запитом (наприклад, вміст HTML-форми) або документ, що відправляється у відповідь. Наявність тіла та його розмір визначається стартовим рядком та заголовками HTTP.[19]

Стартовий рядок разом із заголовками повідомлення HTTP називають головою запиту, яке дані - тілом. [19]

Архітектура клієнт-сервер визначає принципи спілкування між комп'ютерами, а правила і взаємодії визначені в протоколі.

Висновки до розділу: у першому розділі було розкрито теоретичну основу тестування ПЗ. Розкрили поняття тестування у широкому та вузькому сенсі, описали та структурували види тестування. Описали поняття, переваги, недоліки та методи тестування Web-додатку. Також охарактеризували архітектуру «клієнт-сервер», функції, які виконуються на стороні клієнта та

сервера, описали структуру HTTP запитів. Наступним кроком буде опис та вибір інструментів для автоматизованого тестування.

2 ІНСТРУМЕНТИ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ WEB-ДОДАТКІВ

2.1 Критерії вибору інструментів

За останні кілька років з'явилося не менше десятка абсолютно нових інструментів автоматизації тестування інтерфейсу користувача. Оскільки кожен інструмент має власну спрямованість і стратегію, важко зрозуміти, з чого почати.

Критерії для вибору інструменту автоматизованого тестування:

- Простота використання. Інструмент автоматизації повинен бути простим у використанні людьми з різних технічних та нетехнічних кіл. Він має бути інтуїтивно зрозумілим, а тести мають бути простими для написання, розуміння та виконання у кількох проектах;
- Підтримка різних видів додатків. Інструмент повинен надавати можливість писати окремий тест у IDE та запускати його на різних мобільних пристроях, у браузерах та версіях ОС без проблем та паралельно.
- Інтеграція CI/CD. У зв'язку зі зростанням популярності таких гнучких процесів розробки, як Devops, DevSecOps, ATDD та BDD, важливо забезпечити безшовну інтеграцію CI/CD. Необхідно проводити автоматичні тести при кожній реєстрації коду, а також періодично протягом дня, щоб переконатися, що критичні функції системи все ще працюють належним чином і новий змішаний код не порушує інші функції програми.
- Звітність. Обраний інструмент повинен мати різні способи звітування та інструментальні панелі, щоб швидко показати, які тести пройшли та які ні, скільки тестів було проведено, стан автоматизованих пакетів і, звичайно ж, можливість експортувати ці результати.

2.2 Актуальні інструменти у 2023 році

Автоматизовані засоби тестування - це програмні інструменти, які сприяють процесу тестування настільних, веб- та мобільних додатків. Вони забезпечують рішення для автоматизації процесу тестування та мають безліч функцій для тестування графічного інтерфейсу, продуктивності, навантаження та API.

Декілька популярних засобів автоматизованого тестування:

1) Katalon Studio.
2) Selenium.
3) TestingWhiz.

Розглянемо кожен з цих інструментів, детально оцінимо їхні параметри, переваги та недоліки.

2.3 Характеристика Katalon Studio

Katalon Studio - це комплексна платформа для автоматизованого тестування веб-додатків, API, мобільних та настільних (Windows) додатків з використанням низько кодового підходу. Знижуючи вимоги до програмування, Katalon Studio був розроблений з метою допомогти командам прискорити процес розробки та забезпечити більш продуктивний контроль якості. [20]

Katalon Studio є першим інструментом, представленим компанією Katalon Inc у 2015 році. До екосистеми Katalon було додано ще два компоненти:

- Katalon Recorder: розширення для автоматизації браузера, яке дозволяє створювати та запускати тести в Firefox, Edge і Chrome.
- Katalon TestOps: платформа для оркестрування тестів, яка централізує всі процеси планування та управління тестуванням

для оптимізації роботи в умовах DevOps та співпраці між командами. [20]

Деякі ключові функції Katalon Studio:

- Просте та легке впровадження. Координований пакет для впровадження Katalon Studio майже містить все необхідне для розгортання цього потужного і значущого інструменту автоматизації тестування.
- Просте та швидке налаштування. Разом з простою установкою, цей інструмент також спрощує налаштування відповідно до його середовища. В якості тестувальника ви можете швидко запустити перший скрипт за допомогою попередньо побудованих тестових скриптів та шаблонів, таких як ключові бібліотеки та репозиторії об'єктів.
- Швидкі та кращі результати. Вбудовані шаблони поставляються з лаконічними посібниками, які допомагають швидко побудувати та запустити автоматизовані скрипти тестування. Незалежно від налаштування проекту, створення тесту, виконання, генерації звітів або підтримки, ви можете виконувати кожен крок з ефективністю та швидкістю.
- Гнучкі режими. Цей інструмент також пропонує різноманітні гнучкі режими. Якщо ви новий тестувальник, ви можете використовувати записи та ключові слова для побудови автоматизованих тестів. Однак, якщо ви експерт у цій галузі, ви можете використовувати повноцінне середовище розробки для створення складних скриптів.
- Легкість використання. Одна з найкращих рис цього інструменту - його простота використання. Навіть якщо у вас мінімальний досвід програмування, ви все ж зможете легко ним користуватися.

- Підтримка різних операційних систем. Katalon Studio легко підтримує різні платформи, такі як OS X 10.5+ та Windows 32 і 64 (7, 8 і 10). [21]

2.4 Характеристика Selenium

Selenium - це відкритий та переносний автоматизований інструмент для тестування веб-додатків. Він має можливості працювати з різними браузерами та операційними системами. Selenium - це не просто один інструмент, а набір інструментів, які допомагають тестувальникам автоматизувати веб-додатки більш ефективно. [22]

Більшість програмістів і розробників, які створюють веб-додатки і періодично тестують їх, використовують Selenium. Однією з найбільших переваг Selenium, яка зробила його таким популярним, є його гнучкість. Будь-яка особа, яка створює веб-програми, може використовувати Selenium для тестування коду та додатків. Крім того, фахівці можуть влагоджувати та проводити тести на візуальну регресію відповідно до вимог веб-сайту або коду. [23]

У більшості організацій це обов'язок інженерів з контролю якості (QA) тестувати веб-додатки з використанням Selenium. Вони повинні писати скрипти, які допоможуть максимізувати точність і охоплення тестами для внесення змін в проект та підтримки інфраструктури тестування. [23]

Інженери QA відповідають за розробку наборів тестів, які можуть виявляти помилки, за допомогою яких вони можуть повідомляти стейк холдерів про встановлені стандарти для проекту. Основна мета інженерів QA - забезпечити ефективність і охоплення тестами, а також збільшити продуктивність. [23]

2.5 Характеристика TestingWhiz

TestingWhiz - це інструмент для автоматизованого тестування без написання коду, який надає рішення для автоматизованого тестування глобальних підприємств і програмних компаній їх програмного забезпечення, веб-сайтів, мобільних додатків, баз даних, хмарних рішень, веб-сервісів та API. [24]

TestingWhiz автоматизує, виконує та керує тестовими сценаріями з легкістю та ефективністю. Він також дозволяє користувачам перевіряти критичну функціональність їх веб-додатків та забезпечувати ефективні та ефективні веб-інтерфейси. [24]

TestingWhiz інтегрується з популярними інструментами, фреймворками, базами даних та платформами, щоб забезпечити безшовну автоматизацію та виконання тестових сценаріїв у різних середовищах і умовах. Завдяки цим інтеграціям TestingWhiz дозволяє забезпечити краще з'єднання між технічною архітектурою вашого додатка і розширити можливості автоматизації. [24]

Деякі зі значних інтеграцій, які доступні в TestingWhiz для забезпечення ефективного досвіду автоматизації для тестувальників і менеджерів з якості, наведені нижче [24]:

- Інструменти управління тестуванням
- Інструменти безперервної інтеграції
- Інструменти відстеження помилок
- Управління контролем версій
- Хмарні платформи для виконання
- Бази даних
- Мобільні платформи

2.6 Вибір програмного забезпечення

З перелічених інструментів тестування програмного забезпечення був обраний Selenium WebDriver, так як це простий API, який може допомогти в автоматизації браузера.

В якості IDE була обрана Microsoft Visual Studio 2022 через вдосконалену продуктивність та швидкодію, покращену підтримку C#, вбудовану підтримку тестування, профілювання та налагодження коду, велику кількість плагінів і розширень для додавання додаткових функцій.

Також Visual Studio 2022 дозволяє розробникам працювати з багатим набором інструментів, які допомагають створювати високоякісні програми швидше і ефективніше. Це інтегроване середовище підходить для розробки різноманітних програмних продуктів, від веб-сайтів до настільних програм і мобільних додатків.

2.7 Вибір мови програмування

Основні фактори, які варто враховувати при виборі мови програмування для проведення тестування, можуть включати:

1) Підтримка тестового фреймворку: Деякі тестові фреймворки можуть бути доступні лише для деяких мов програмування. Тому вибір мови може обмежувати вибір тестового фреймворку.
2) Швидкість виконання.
3) Наявність бібліотек та інструментів.
4) Підтримка для автоматизації тестування.
5) Підтримка для паралельного виконання.
6) Сумісність з іншими інструментами: якщо потрібно інтегрувати тестовий фреймворк з іншими інструментами, такими як система збірки, CI/CD-система, то важливо знайти мову, яка підходить для інтеграції з цими інструментами.
7) Наявність підтримки та активність спільноти.

Система автотестів була реалізована на мові C#.

C# — це об’єктно-орієнтована мова, яка підтримує багато функцій, включаючи успадкування, поліморфізм, інкапсуляцію, делегати, лямбда-вирази, атрибути тощо. Основна ідея C# — забезпечити високий рівень безпеки та продуктивності під час розробки програмного забезпечення.

Висновки до розділу: у цьому розділі було сформовано критерії вибору інструментів та мови програмування для створення автотестів. Розглянули популярні інструменти у 2023 році, описали їх переваги на недоліки. За підсумком, для створення системи автотестів було обрано Selenium Webdriver та Visual Studio 2022 в якості IDE. В якості мови програмування був обраний C#.

3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

3.1 Опис веб-додатка

На даному етапі необхідно описати web-додаток, щоб на його основі розробити тест-план з використанням документації. Після написання тест-кейсів наступним етапом буде перевірка функціональності сайту та його основних компонентів.

Для автоматизованого тестування я обрав інтернет магазин одягу: https://arber.ua/.

Arber – сайт, який містить основні функції інтернет-магазину. Використовуючи сайт, користувач може переглядати каталог товарів, шукати товари, дивитися опис товарів, додавати в кошик / вибраного, оформляти заявку.

3.2 Відображення процесу проходження документації

Після встановлення усіх необхідних додатків, переходимо до опису документації.

Перший етап полягає в складанні тест-плану і чек-листа.

Таблиця 3.1 – Історія створення тест-плану

Дата	Версія	Опис дії	Автор
15.05.2023	1.0	Створення	Пунтус Є.В.

В описі тест-плану будуть присутні такі пункти:

- Вихідні дані.
- Види тестування.
- Операційні системи, браузери.

- Функціонал додатка.
- Критерії початку тестування.
- Критерії виходу з тестування.
- Характеристики обладнання.
- Кінцеві результати.

1) Вихідні дані

Arber.com – сайт, який містить основні функції інтернет-магазину. Використовуючи сайт, користувач може переглядати каталог товарів, шукати товари, дивитися опис товарів, додавати в кошик / вибраного.

2) Види тестування

Тестування буде проводитися вручну і за допомогою автотестів, методом «неформального» тестування (adhoc testing) з позиції кінцевого користувача сайту.

3) Операційні системи, браузери

ОС, яка буде використовуватися під час тестування:

• Windows 11

Браузер, який буде використовуватися під час тестування:

• Google Chrome 113.0.5672.93.

4) Функціонал додатка

Веб-сайт повинен забезпечити користувачів можливостями для перегляду каталогів товарів, здійснення пошуку, ознайомлення з описом товарів, додавання їх до кошика або списку бажань.

5) Критерії початку тестування

Інструменти для проведення тестування та мова програмування обрані та затверджені, додаткові фреймворки та додатки встановлені та підключені до проекту.

6) Критерії виходу із тестування

Основний функціонал веб-додатку успішно протестований, критичні помилки та баги відсутні, всі тести пройдені успішно.

7) Характеристики обладнання

Тестування проводилось на комп’ютері з наступними характеристиками:

- Процесор: AMD Ryzen 5 4600h, CPU @ 3GHz/4GHz
- Встановлена пам’ять (ОЗП): 8 Гб.
- Відеокарта: Nvidia GeForce 1650Ti.
- Тип системи: 64-розрядна.
- Операційна система: Windows 11.

8) Кінцеві результати

Результатом процесу тестування будуть наступні матеріали:

• підсумок тестування відносно загального стану, що дає розробникам і менеджерам даного продукту картину відносно коректності роботи сайту;

• аналіз результатів тестування поточного покриття.

3.3 Автоматизоване тестування веб-додатку

У додатку Visual Studio створюємо новий тестовий проект NUnit. Проект створений під назвою «store_test» (рис. 3.1).

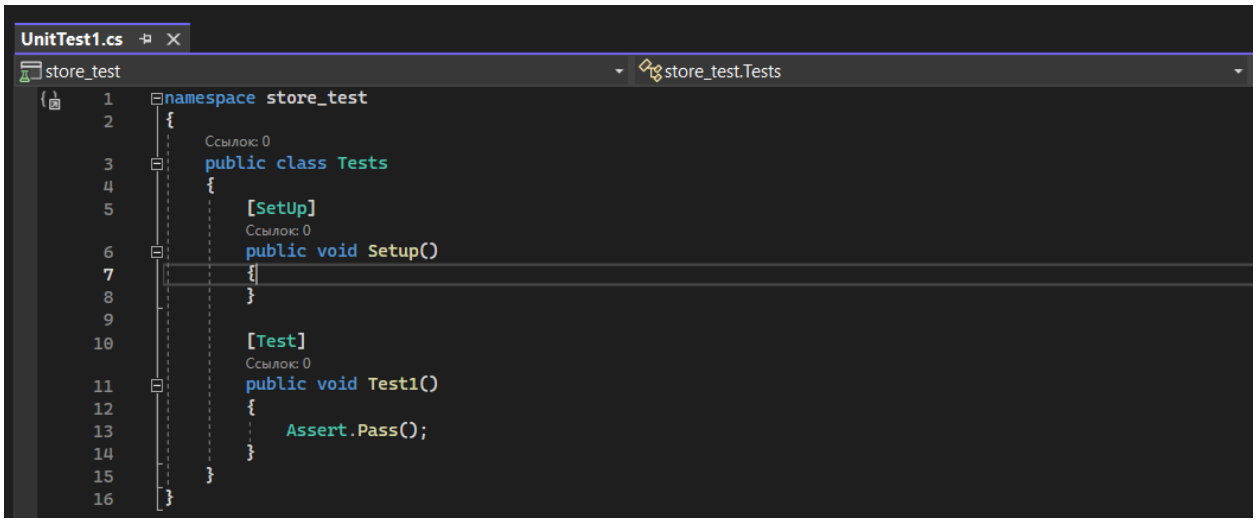

Рисунок 3.1 – Створення проекту

Так виглядає початкова сторінка створеного проекту.

namespace store_test – назва простору імен, в якому будуть знаходитись певні класи та методи.

public class Tests – основний клас з модифікатором доступу public з двома публічними методами.

public void Setup – метод, який буде запускатись перед виконанням тестів.

Далі будуть знаходитись основні методи з тестами.

Для того, щоб почати працювати з Selenium WebDriver, спочатку його потрібно підключити до нашого проекту. У Visual Studio це можна зробити за допомогою менеджеру пакетів. Нажимаємо на вкладку «Проект» > «Управління пакетами NuGet» (рис. 3.2).

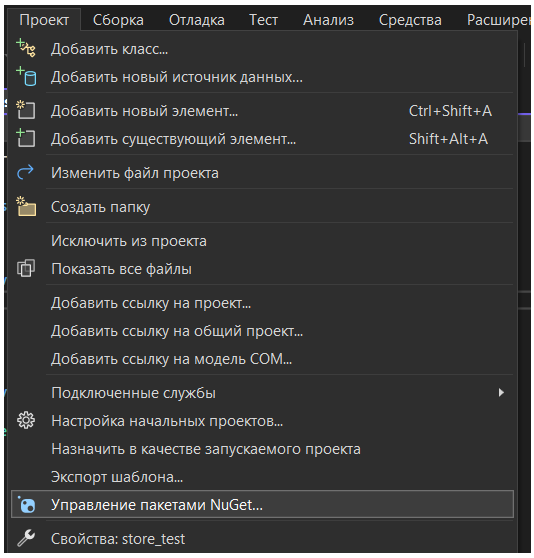

Рисунок 3.2 – Запуск менеджера пакетів

Після цього відкривається відповідне вікно, в якому ми обираємо які пакети підключити до проекту. У конкретно нашому випадку, нам потрібен «Selenium.WebDriver» та «Selenium.Chrome.WebDriver».

Також для роботи потрібно встановити Chrome.WebDriver. Для цього завантажуємо останню версію драйверу та розпаковуємо в будь-яке зручне місце на комп’ютері. Далі нам потрібно підключити його до нашого проекту. Для цього в класі створюємо приватну змінну *driver* типу *IWebDriver* та в методі *Setup()* прописуємо наступний код:

var options = new ChromeOptions().BinaryLocation = @"D:\Diploma\chromedriver_win32\chromedriver.exe";

driver = new ChromeDriver(options);

//для запуску браузера необхідно ініціалізувати об'єкт драйвера

Також використаємо методи відкриття нашої сторінки «https://arber.ua/?sl=uk» на повний екран:

driver.Manage().Window.Maximize();

Цей приклад викликає GoToUrl метод для переходу до веб-сторінки за адресою https://arber.ua/?sl=uk, передаючи в якості аргументу URL адресу сторінки:

driver.Navigate().GoToUrl("https://arber.ua/?sl=uk");

Головна сторінка нашого веб-додатку (рис. 3.3):

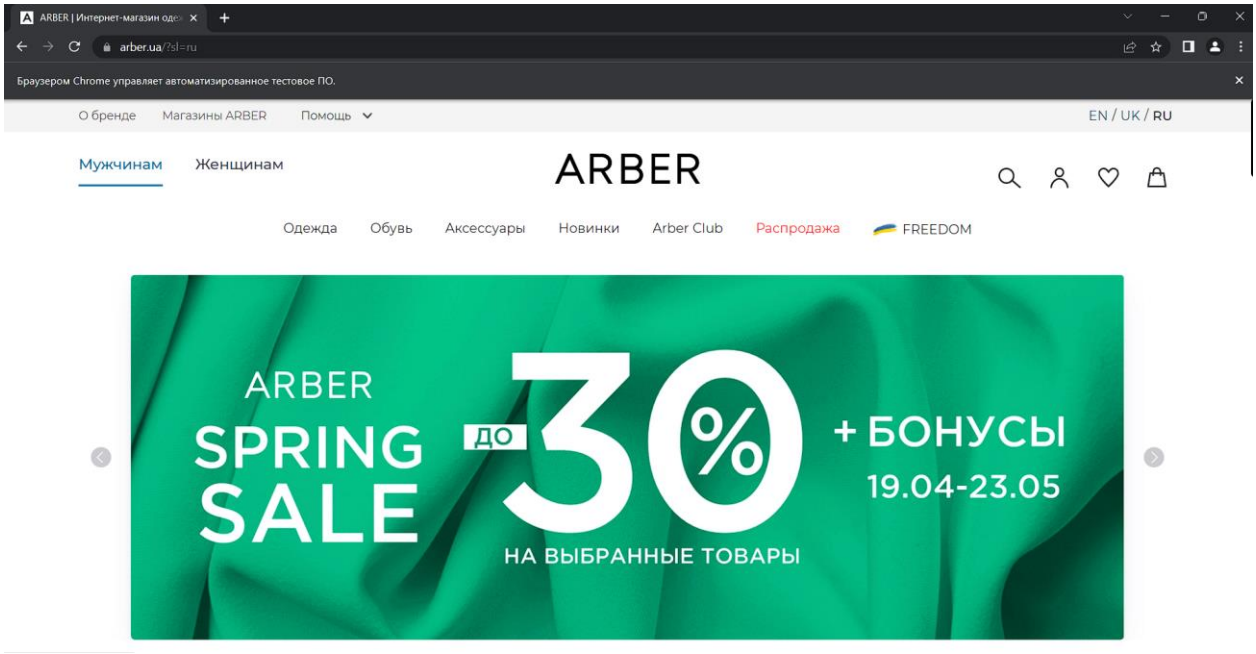

Рисунок 3.3 – Головна сторінка веб-додатку

Опишемо позитивний сценарій тестування у вигляді тест-кейсу.

Опис позитивного тест-кейсу №1

Опис тесту: авторизація користувача в систему.

Передумова: Створено обліковий запис для входу в особистий кабінет (логін/пошта – yehor.puntus@gmail.com, пароль – FeX4rRqzp2CzgD).

Кроки:

1) Перейти на посилання https://arber.ua/?sl=uk;
2) Залогінитись використовуючи дані облікового запису;
3) Переглянути Main menu на Головній сторінці сайту.

Очікуваний результат: Відкривається головна сторінка авторизованого користувача.

Лістинг виконання автотесту (додаток А).

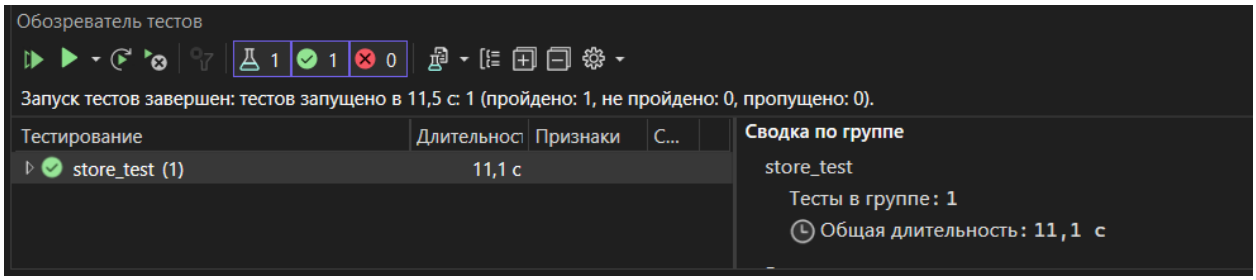

Рисунок 3.4 - Результат успішного виконання тесту

Забезпечення безпека та конфіденційності даних користувачів веб-додатку - є однією з найважливіших задач тестування. Для цього необхідно перевірити, що користувач зможе зайти в особовий кабінет лише використовуючи валідні дані авторизації (в нашому випадку пошта та пароль). Для відтворення наступного тест-кейсу, використаємо код з попереднього та виконаємо тест з не дійсним паролем.

Опис негативного тест-кейсу №2

Опис тесту: авторизація користувача в систему, використовуючи невірний пароль.

Передумова: Створено обліковий запис для входу в особистий кабінет (логін/пошта – yehor.puntus@gmail.com, пароль – FeX4rRqzp2CzgD).

Кроки:

1) Перейти на посилання https://arber.ua/?sl=uk;
2) Ввести в поле «E-mail» дійсний e-mail «yehor.puntus@gmail.com»
3) Ввести в поле «Пароль» не дійсний пароль.
4) Натиснути кнопку «Увійти».

Очікуваний результат: Відкривається спливаюче повідомлення про помилку.

Переглянемо результат у вигляді скріншота екрана під час виконання негативного тест-кейсу з введенням не валідного пароля та виведення повідомлення про помилку (рис. 3.5).

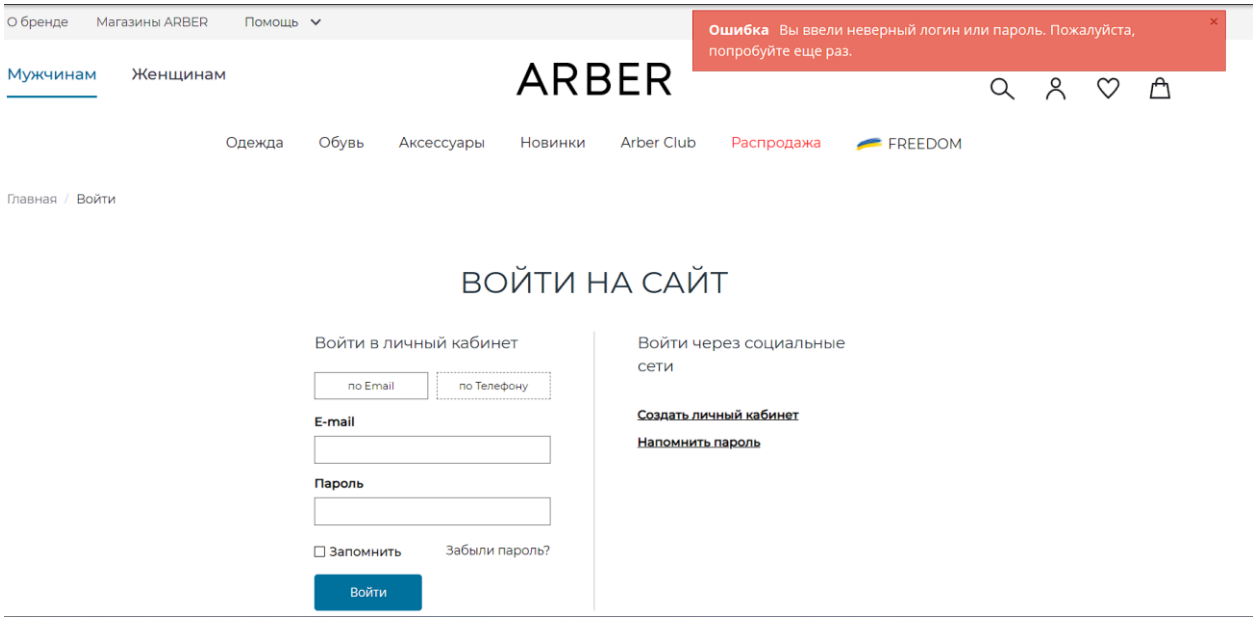

Рисунок 3.5 - Сторінка авторизації користувача з повідомлення про помилку

На сторінці інтернет-магазину найбільш використовуваним елементом є поле "Пошук" та фільтри. Введемо в поле "Пошук" категорію "Взуття" та виберемо "Чоловікам – чоловічий одяг" зі списку. Потім ми скористаємось фільтром категорії "Кольори" і виберемо "Білий" колір, щоб переглянути результат.

Опис позитивного тест-кейсу №3

Опис тесту: пошук елемента по фільтру.

Кроки:

1) Перейти на посилання https://arber.ua/?sl=uk;
2) Натиснути на поле пошуку;
3) Ввести в поле «Взуття»;
4) Обрати зі списку «Чоловікам – чоловічий одяг»;

5) Обрати у фільтрі кольорів «Білий».

Очікуваний результат: Відображення всіх товарів категорії «Чоловіче взуття» з кольором «Білий».

Лістинг тесту наданий в додатку А. Результати виконання тестового тест-кейсу №2 (рис. 3.6 та 3.7).

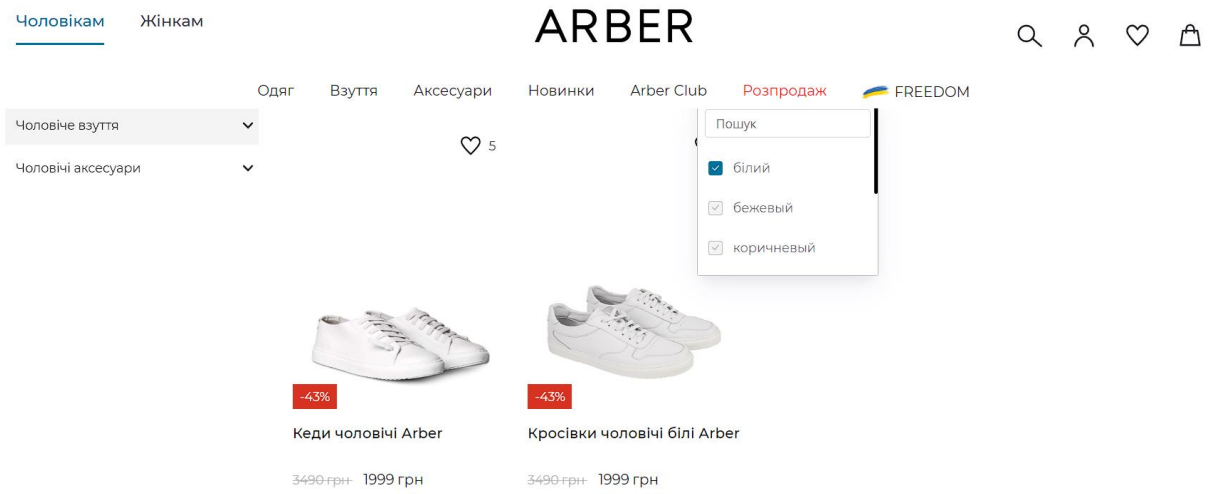

Рисунок 3.6 - Результати виконання тест-кейсу №2

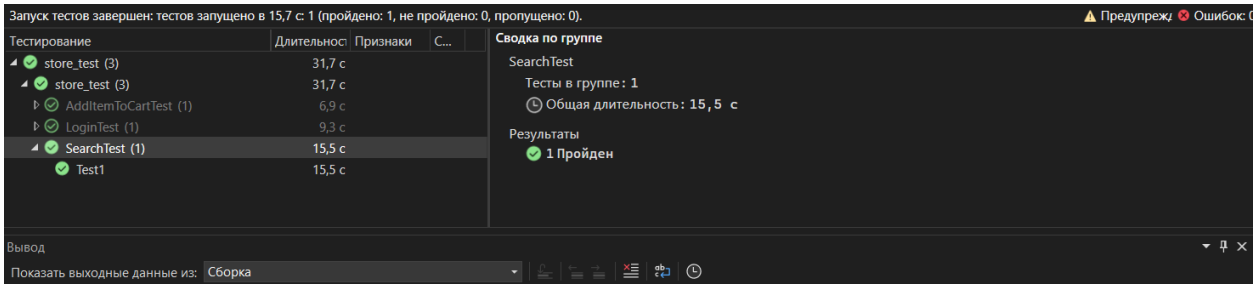

Рисунок 3.7 – Результат виконання тесту в Visual Studio

Наступним за кількістю використання елементом на сторінці є «Кошик», тому третій тест буде перевіряти додавання товару до кошика. Переходимо на сторінку товару. Підбираємо розмір і додаємо у кошик. Також для відтворення результату додаємо умову: якщо товар у корзині = 0, то в консолі виконання тесту з’явиться повідомлення "Товар не додався до кошика".

Опис позитивного тест-кейсу №4

Опис тесту: додавання та перевірка наявності товару у корзині

Кроки:

1) Перейти на сторінку товару.
2) Натиснути на кнопку «У кошик».
3) Натиснути на іконку кошику.
4) Перевірити, що кошик не пустий.

Очікуваний результат: товар доданий до кошика.

Лістинг тесту наданий в додатку А. Результати виконання тестового тест-кейсу №3 (рис. 3.7).

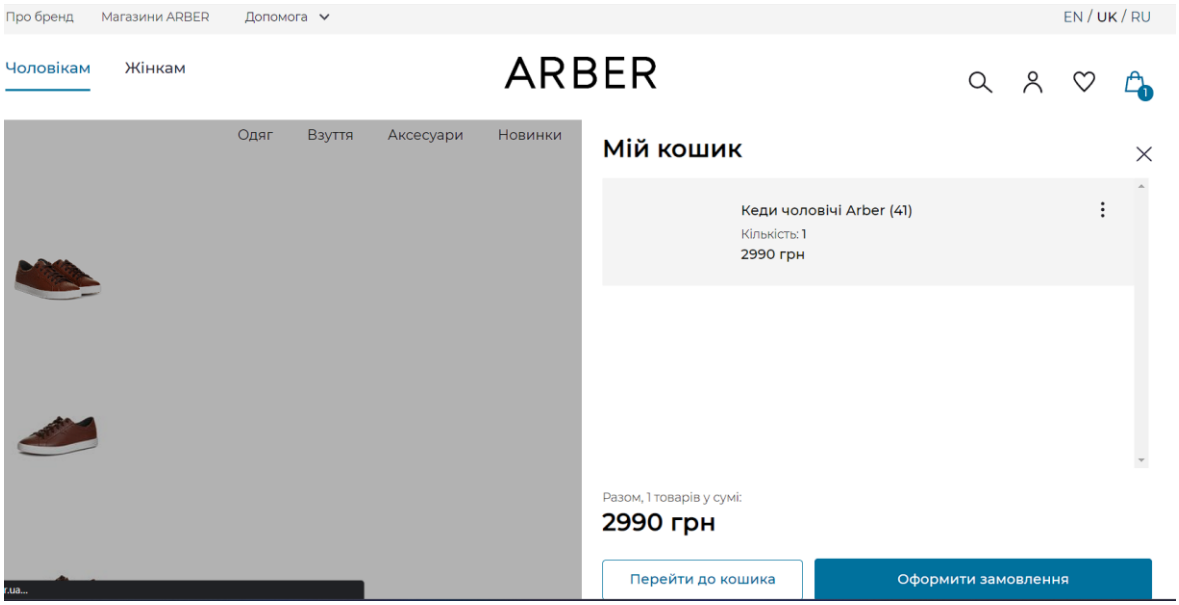

Рисунок 3.8 – Результат виконання тест-кейсу №3

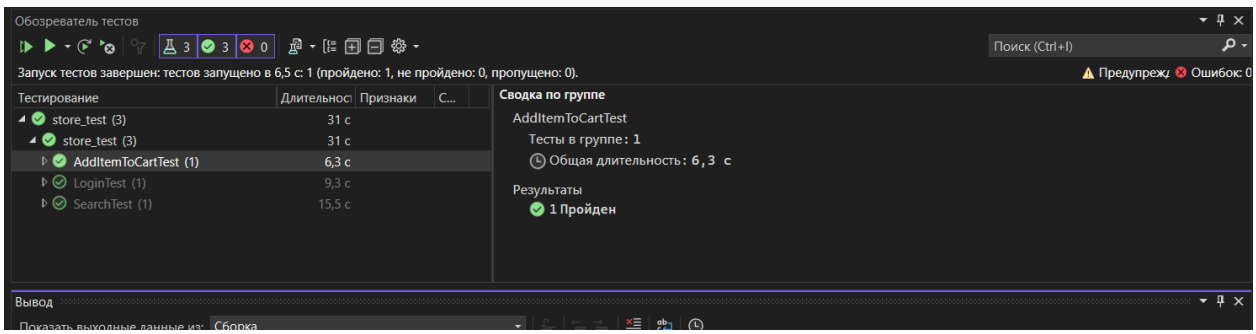

Рисунок 3.9 – Результат виконання тесту в Visual Studio

3.4 Аналіз отриманих результатів

Після закінчення процесу тестування ПЗ необхідно проаналізувати результати та зробити висновки щодо коректності роботи веб-додатка. Загальний час проходження тестів відтворений у вигляді таблиці (Таблиця 3.2).

Таблиця 3.2 – Загальні результати тестування

№ Тест кейса	Час виконання	Результат виконання
1	8.7с	Успішно пройдений
2	8.6с	Успішно пройдений
3	10.6с	Успішно пройдений
4	8.6с	Успішно пройдений

Тест кейс №1 - Позитивний тест авторизації користувача.

У цьому тесті ми перевіряли функціонал авторизації користувача на сайті. Він включав перехід на головну сторінку сайту, введення облікових даних та перевірку, що після успішної авторизації відкривається головна сторінка авторизованого користувача. Результати тесту показали, що функціонал авторизації працює належним чином, оскільки користувач успішно авторизувався і перейшов на головну сторінку. Це підтверджує правильну роботу механізму авторизації і дозволяє користувачеві отримувати доступ до особистого кабінету.

Тест кейс №2 - Негативний тест авторизації користувача з невірним паролем.

У цьому тесті ми перевіряли, як система поводиться, коли користувач намагається авторизуватись, використовуючи невірний пароль. Ми використали дійсну електронну адресу, але ввели неправильний пароль. Очікувалося, що після спроби входу з невірним паролем, система повинна показати спливаюче повідомлення про помилку.

Результати тесту показали, що функціонал авторизації працює правильно. Після спроби авторизуватись з невірним паролем, система відобразила спливаюче повідомлення про помилку, що вказує на неправильні облікові дані.

Тест кейс №3 - Позитивний тест пошуку елемента по фільтру.

У цьому тесті ми перевіряли функціонал пошуку товарів за фільтром на сайті. Користувачу було необхідно ввести певні параметри пошуку (категорію та колір) і переконатися, що відображаються всі товари, які відповідають цим параметрам. Результати тесту показали, що функціонал пошуку за фільтром працює належним чином, оскільки були відображені всі товари категорії "Чоловіче взуття" з кольором "Білий". Це свідчить про правильну роботу пошукового механізму, який допомагає користувачам знайти потрібні товари на сайті.

Тест кейс №4 - Позитивний тест додавання та перевірки наявності товару у кошику.

У цьому тесті ми перевіряли функціонал додавання товару до кошика та перевірку наявності товару в кошику. Користувачу потрібно було перейти на сторінку товару, натиснути на кнопку "У кошик" і перевірити, що товар відображається в кошику. Результати тесту показали, що функціонал додавання товару до кошика та перевірки наявності товару в кошику працює належним чином, оскільки товар був успішно доданий і відображається в кошику. Це свідчить про правильну роботу механізму управління кошиком, який дозволяє користувачам зберігати та контролювати обрані товари.

Висновки до розділу: всі тести успішно пройдені, і жодних помилок або багів не виявлено. Це свідчить про те, що система функціонує правильно і відповідає вимогам. Результати тестування дають підставу стверджувати, що розроблений функціонал відповідає очікуванням та готовий до використання користувачами.

ВИСНОВКИ

У даному дипломному проекті була проведена розробка та реалізація системи автоматизованого тестування веб-додатків. У першому розділі було розкрито теоретичну основу тестування ПЗ. Розкрили поняття тестування у широкому та вузькому сенсі, описали та структурували види тестування. З'ясували, що будь-яке тестування має на увазі дві діючі особи: суб'єкт і об'єкт тестування. Суб'єктом тестування (тобто джерелом активності) виступає «тестувальник» (tester) – спеціально призначена особа, в обов'язки якої входить виконання тестування. Під об'єктом тестування розуміється досліджувана система. При цьому в ролі тестованої системи може виступати як весь продукт, так і окремі його модулі, складові частини.

Описали поняття, переваги, недоліки та методи тестування Web-додатку. Також охарактеризували архітектуру «клієнт-сервер», функції, які виконуються на стороні клієнта та сервера, описали структуру HTTP запитів.

Проаналізували найпопулярніші інструменти з тестування програмного забезпечення на ринку, описали їх переваги та недоліки, що дозволило обрати оптимальну програму для створення системи автоматизованого тестування веб-додатку.

На основі усієї зібраної інформації, була спроектована система автоматизованого тестування веб-додатку, яка складається з чотирьох тест-кейсів та їх програмних реалізацій.

Система показала, що всі тести успішно пройдені, і жодних помилок або багів не виявлено. Це свідчить про те, що система функціонує правильно і відповідає вимогам. Результати тестування дають підставу стверджувати, що розроблений функціонал відповідає очікуванням та готовий до використання користувачами.

ПЕРЕЛІК ПОСИЛАНЬ

1) IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, 1990. 84p.
2) Авраменко А.С., Авраменко В.С., Косенюк Г.В., Тестування Програмного Забезпечення, навчальний посібник, Черкаси, 2017. 284с. URL: http://eprints.cdu.edu.ua/1482/1/testyvan.pdf (дата звернення 21.04.2023).
3) Ravindra Savaram, Manual Testing Tutorial for Beginners, електронний ресурс. 2023. URL: https://mindmajix.com/manual-testing-tutorial#types-of-manual-testing (дата звернення 21.04.2023).
4) Що таке тест-план, для чого він потрібен і з чого складається, QATestLab: електронний ресурс. 2023. URL: https://training.qatestlab.com/blog/technical-articles/test-plan/ (дата звернення 21.04.2023).
5) Max Rehkopf, Automated software testing: електронний ресурс. 2023. URL: https://www.atlassian.com/continuous-delivery/software-testing/automated-testing#:~:text=What%20is%20automated%20testing%3F,include%20automated%20testing%20from%20inception (22.04.2023).
6) Kate Brush, Test Case definition: електронний ресурс. 2023, URL: https://www.techtarget.com/searchsoftwarequality/definition/test-case (дата звернення 22.04.2023).
7) Thomas Hamilton, What is Functional Testing? Types & Examples: електронний ресурс. 2023. URL: https://www.guru99.com/functional-testing.html (дата звернення 23.04.2023).
8) Glenford J. Myers, Tom Badgett, Corey Sandler, The Art Of Software Testing: посібник. New Jersey, 2012. 254p.

9) Т.М. Коротун, Моделі і методи тестування: електронний ресурс, 2007. 9с. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/289/%D0%9A%D0%BE%D1%80%D0%BE%D1%82%D1%83%D0%BD_%231.pdf?sequence=1 (дата звернення 23.04.2023).

10) Usability Testing, Usability.gov: електронний ресурс. 2023. URL: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html#:~:text=Usability%20testing%20refers%20to%20evaluating,watch%2C%20listen%20and%20takes%20notes (дата звернення 25.04.2023).

11) Тестування UI (інтерфейсу користувача), Wezom: електронний ресурс. 2021. URL: https://wezom.com.ua/ua/blog/testing-ui-user-interface (дата звернення 25.04.2023).

12) Madhu Anjanappa, The Six Principles of Security Testing: електронний ресурс. 2017. URL: https://blog.trigent.com/the-six-principles-of-security-testing/ (дата звернення 25.04.2023).

13) Thomas Hamilton, What is Localization Testing? Example Test Cases & Checklist: електронний ресурс. 2023. URL: https://www.guru99.com/localization-testing.html (дата звернення 27.04.2023).

14) Автоматизуємо тестування: коли, навіщо і кому це потрібно: електронний ресурс. 2023. URL: https://newline.tech/test-automation-when-why-and-who-needs-it_uk/ (дата звернення 28.04.2023).

15) Автоматизоване тестування. Qalight.ua: електронний ресурс. 2023. URL: https://qalight.ua/baza-znaniy/avtomatizovane-testuvannya/ (дата звернення 28.04.2023).

16) Що таке веб додаток? Різниця між сайтом, веб-додатком, spa і pwa. Webcase.com.ua: електронний ресурс. 2022. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення 29.04.2023).

17) Тестування веб-проектів: основні етапи та поради. Qalight.ua: електронний ресурс. 2023. URL: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/ (дата звернення 29.04.2023).

18) Клієнт-серверна архітектура, QATestLab: електронний ресурс. 2020. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення 29.04.2023).

19) HTTP Messages, MDN web docs: електронний ресурс. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages (дата звернення 30.04.2023).

20) What Is Katalon Studio? Overview & Tour Of Features, Katalon Team, 2023. URL: https://theqalead.com/test-management/katalon-studio-overview/ (дата звернення 30.04.2023).

21) Madhuri Yerukala, What is Katalon Studio - Complete Tutorial Guide: електронний ресурс. 2023. URL: https://mindmajix.com/katalon-studio-tutorial (дата звернення 01.05.2023).

22) Selenium – Overview, Tutorialspoint: електронний ресурс. 2023. URL: https://www.tutorialspoint.com/selenium/selenium_overview.htm (дата звернення 01.05.2023).

23) Prabhpreet Kaur Bhamra, What is Selenium? Introduction to Selenium Automation Testing: електронний ресурс. 2023. URL: https://intellipaat.com/blog/tutorial/selenium-tutorial/introduction/?US (дата звернення 02.05.2023).

24) About TestingWhiz, Cygnet Infotech: електронний ресурс. 2023. URL: https://www.testing-whiz.com/user-manual/introducing-testingwhiz/ (дата звернення 02.05.2023).

ДОДАТОК А

Лістинг тест кейсу №1

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace store_test
{
    public class LoginTest
    {
        private IWebDriver driver;
        private readonly string url = "https://arber.ua/?sl=uk";
        private readonly string username =
"yehor.puntus@gmail.com";
        private readonly string password = "FeX4rRqzp2CzgD";

        private readonly By _profileButtonLocator =
By.XPath("//div[@class=\"header-icon user\"]");
        private readonly By _signInButtonLocator =
By.XPath("//a[text()='Увійти']");
        private readonly By _loginButtonLocator =
By.XPath("//input[@id=\"login_main_login\"]");
        private readonly By _passwordInputLocator =
By.XPath("//input[@id=\"psw_main_login\"]");
        private readonly By _submitButtonLocator =
By.XPath("//form[@name='main_login_form']/child::button");
        private readonly By _successAlertLocator =
By.XPath("//strong[text()=\"Сповіщення\"]");

        [SetUp]
        public void Setup()
        {
            var options = new ChromeOptions().BinaryLocation =
@"D:\Diploma\chromedriver_win32\chromedriver.exe";
            driver = new ChromeDriver(options);

            driver.Navigate().GoToUrl(url);
            driver.Manage().Window.Maximize();
        }

        [Test]
        public void LoginTest1()
        {
            driver.FindElement(_profileButtonLocator).Click();
            driver.FindElement(_signInButtonLocator).Click();
```

```
            driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(10);


driver.FindElement(_loginButtonLocator).SendKeys(username);

driver.FindElement(_passwordInputLocator).SendKeys(password);

            driver.FindElement(_submitButtonLocator).Click();


Assert.IsTrue(IsElementDisplayed(_successAlertLocator));
        }

        private bool IsElementDisplayed(By locator)
        {
            try
            {
                return driver.FindElement(locator).Displayed;
            }
            catch (NoSuchElementException)
            {
                return false;
            }
        }

        [TearDown]
        public void TearDown()
        {
            driver.Quit();
        }
    }
}
```

Лістинг тест кейсу №3

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Interactions;

namespace store_test
{
    public class SearchTest
    {
        private IWebDriver driver;
        private Actions _actions;

        private readonly By _searchInputLocator =
By.XPath("//input[@title='Пошук']");
        private readonly By _itemToSelectLocator =
By.XPath("//a[@href='https://arber.ua/cholovkam-uk/vzuttya/']");
        private readonly By _searchIconLocator =
By.CssSelector("div.pos-r > button");
```

```
        private readonly By _colorDropdownLocator =
By.CssSelector("#sw_content_32_17 > span");
        private readonly By _whiteColorOptionLocator =
By.CssSelector("#content_32_17 > li:nth-child(2) > label");
        private readonly By _itemsContainerLocator =
By.XPath("//div[@id=\"read_more_1\"]/div[1]");

        [SetUp]
        public void Setup()
        {
            var options = new ChromeOptions().BinaryLocation =
@"D:\Diploma\chromedriver_win32\chromedriver.exe";
            driver = new ChromeDriver(options);
            _actions = new Actions(driver);

            driver.Navigate().GoToUrl("https://arber.ua/");
            driver.Manage().Window.Maximize();
        }

        [Test]
        public void Test1()
        {
            var search = driver.FindElement(_searchIconLocator);
            _actions.MoveToElement(search).Perform();


driver.FindElement(_searchInputLocator).SendKeys("Взуття
чоловіче");

            driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(10);

            driver.FindElement(_itemToSelectLocator).Click();
            driver.FindElement(_colorDropdownLocator).Click();

driver.FindElement(_whiteColorOptionLocator).Click();

            var itemContainer =
driver.FindElements(_itemsContainerLocator);
            Assert.IsTrue(itemContainer.Count > 0, "Тест не
пройдений: Продукти, що відповідають критеріям фільтра, не
відображаються.");
        }
    }
}
```

Лістинг тест-кейсу №4

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace store_test
```

```
{
    public class AddItemToCartTest
    {
        private IWebDriver driver;

        private readonly string url = "https://arber.ua/kedi-
cholovch-arber-uk-6-uk-uk/";

        private readonly By _addToCartButton =
By.CssSelector("#button_cart_234145");
        private readonly By _continueShoppingButton =
By.XPath("//a[text()='Продовжити покупки']");
        private readonly By _cartItems = By.XPath("//div[@class
= 'arber_header-popup-loop']");

        [SetUp]
        public void Setup()
        {
            var options = new ChromeOptions().BinaryLocation =
@"D:\Diploma\chromedriver_win32\chromedriver.exe";
            driver = new ChromeDriver(options);

            driver.Navigate().GoToUrl(url);
            driver.Manage().Window.Maximize();
        }

        [Test]
        public void Test1()
        {
            var addToCartButton =
driver.FindElement(_addToCartButton);
            addToCartButton.Click();

            driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(10);
            var continueShoppingButton =
driver.FindElement(_continueShoppingButton);
            continueShoppingButton.Click();

            var cartItems = driver.FindElements(_cartItems);

            Assert.IsTrue(cartItems.Count > 0, "Тест не
пройдений: видані товари не підпадають під критерії пошуку");
        }

        [TearDown]
        public void TearDown()
        {
            driver.Quit();
        }
    }
}
```