Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп’ютерних систем та мереж

Пояснювальна записка
до кваліфікаційної роботи бакалавра
за спеціальністю 123 «Комп’ютерна інженерія»

на тему: ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ WEB-ДОДАТКІВ

Проектував _____ Є. В. Сердюк

Керівник роботи _____ Ю.О. Кумченко

Нормоконтроль _____ Д. І. Кузнєцов

Завідувач кафедри _____ А. І. Купін

Кривий Ріг
2024

Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Ступінь вищої освіти бакалавр
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії

_____ А. І. Купін

"____" _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____Сердюк Євгеній Володимирович_____
(прізвище, ім'я, по батькові)

1. Тема роботи _____Технології автоматизованого тестування web-додатків_
_____
_____
керівник роботи ____Кумченко Ю.О._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "___"_____ 20__року №__
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи _____
__інформація про тестування, автоматизоване тестування, інструмент_____
автоматизованого тестування._____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____
__Теорія тестування та її види_____
___Переваги та недоліки автоматизованого тестування_____
___Створити систему автотестів_____
_____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)___Презентація PowerPoint_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Пошук підходящої літератури	12.03.24 - 18.03.24	Виконав
2	Розробка структури дипломної роботи та сортування інформації за розділами	20.03.24	Виконав
3	Огляд інформації що стосується тестування	24.03.24 – 28.03.24	Виконав
4	Написання першого розділу	01.04.24- 10.04.24	Виконав
5	Огляд існуючих інструментів	20.04.24	Виконав
6	Написання другого(аналітичного розділу)	23.04.24	Виконав
7	Створення тест плану з описом тест-кейсів	03.05.24	Виконав
8	Створення автотестів	10.05.24	Виконав
9	Написання третього розділу	15.05.24	Виконав
10	Написання пояснювальної записки та загальне оформлення дипломної роботи	20.05.24	Виконав

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 57 сторінки, 25 рисунків, 7 таблиць, 22 використаних джерел.

Об’єкт аналізу – система автоматизованого тестування web-додатків.

Мета – розглянути теорію автоматизації тестування та на її основі створити систему автотестів.

Перший розділ присвячений збору та аналізу інформації про автоматизоване та ручне тестування. У другому розділі були розглянуть популярні сервіси автоматизації тестування web-додатків, були розписані їх особливості, сформовані основні недоліки та переваги кожного з них. У третьому розділі розробили та виконали систему автотестів на базі *Katalon Studio*. Проаналізували отримані результати.

Ключові слова: тестування, автотест, web-додаток, програмне забезпечення, тест-кейс.

Explanatory note

Explanatory note: 57 pages, 25 figures, 7 tables, 22 used sources.

The object of analysis is a system of automated testing of web applications.

The goal is to consider the theory of test automation and create a system of self-tests based on it.

The first section is devoted to the collection and analysis of information on automated and manual testing. In the second section, popular web application testing automation services were considered, their features were described, the main disadvantages and advantages of each of them were formed. In the third section, a self-testing system based on Katalon Studio was developed and implemented. The obtained results were analyzed.

Keywords: testing, autotest, web application, software, test case.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

GUI – графічний інтерфейс користувача.

HTTPS – *HyperText Transfer Protocol Secure.*

IDE – середовище розробки.

XML – розширювана мова розмітки (*Extensible Markup Languege*).

ОС – операційна система.

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер.

ШІ – штучний інтелект;

ВСТУП

У сучасному світі розвитку програмного забезпечення, значна увага приділяється якості та надійності програмних продуктів. Зокрема, веб-додатки вимагають ретельного тестування через їх широке розповсюдження та важливість в повсякденному використанні як в бізнесі, так і в особистому спілкуванні. Технології автоматизованого тестування веб-додатків набувають особливого значення у контексті забезпечення високої продуктивності, безпеки та коректності програмних рішень.

Автоматизація тестування дозволяє значно скоротити час перевірки роботи веб-додатків та підвищити ефективність процесів верифікації. Вона забезпечує більш високу точність виконання тестових сценаріїв, зменшує можливість людської помилки, а також дозволяє автоматично перевіряти велику кількість параметрів інтерфейсу та функціональності додатків. З огляду на це, актуальність дослідження автоматизованих інструментів тестування веб-додатків не може бути переоцінена.

Метою даної дипломної роботи є аналіз сучасних інструментів та методологій автоматизованого тестування, вивчення їх особливостей, можливостей та обмежень, а також розробка рекомендацій щодо їх ефективного використання при розробці веб-додатків. Робота передбачає теоретичний огляд ключових платформ таких, як Selenium, WebDriver, а також практичне застосування цих інструментів для вирішення типових завдань тестування.

Дослідження буде корисним для розробників програмного забезпечення, тестувальників, а також керівників проектів у сфері IT, які прагнуть оптимізувати процеси розробки та підтримки веб-додатків. Результати роботи сприятимуть покращенню якості та ефективності процесів

тестування, а також допоможуть впровадженню кращих практик у сфері автоматизації тестування.

У дипломній роботі буде представлена та проаналізована теорія, що стосується автоматизованого тестування та тестування web-додатків.

Для досягнення поставлених цілей у даному проєкті будуть поставлені та вирішені наступні завдання:

1. Аналіз теорії тестування, їх види.
2. Опис переваг та недоліків автоматизованого тестування.
3. Порівняння інструментів для автоматизації.
4. Оформлення документації *web*-додатку.
5. Створення системи автотестів.
6. Проведення аналізу результатів.

1 РОЗДІЛ СУЧАНА МЕТОДИКА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Поняття тестування ПЗ

Тестування програмного забезпечення (ПЗ) є ключовою частиною процесу розробки ПЗ, спрямоване на виявлення помилок, перевірку відповідності вимогам та забезпечення якості кінцевого продукту. Цей процес включає в себе різні методи та підходи, які дозволяють систематично оцінити стійкість, безпеку, та коректність роботи ПЗ.

Тестування ПЗ можна визначити як активність, що включає в себе планування, створення тестових сценаріїв, виконання цих сценаріїв, аналіз отриманих результатів та оцінку того, наскільки ПЗ відповідає визначеним вимогам і очікуванням зацікавлених сторін. Ця діяльність допомагає ідентифікувати вади в ПЗ до його запуску в експлуатацію.

Основні цілі тестування ПЗ включають:

1. Виявлення помилок: Забезпечення функціональної коректності ПЗ шляхом ідентифікації вад.

2. Перевірка відповідності вимогам: Забезпечення того, що ПЗ відповідає всім зазначеним бізнес-та технічним вимогам.

3. Забезпечення якості: Оцінка загальної якості ПЗ та його компонентів.

4. Підтвердження надійності: Перевірка стійкості та надійності ПЗ у різних умовах.[1]

1.2. Типи тестування ПЗ

Ручне тестування (Manual Testing) – цей метод передбачає взаємодію тестувальника з програмним забезпеченням без використання автоматичних

інструментів. Тестувальник виступає в ролі звичайного користувача, детально перевіряючи всі аспекти продукту на наявність помилок та недоліків. Хоча цей метод є більш часомістким порівняно з автоматизованим тестуванням, він може бути більш економічно вигідним у довгостроковій перспективі.

Ручне тестування може включати такі види:

Модульне тестування: Перевірка окремих компонентів або модулів програми.

Інтеграційне тестування: Тестування взаємодій між модулями програми.

Системне тестування: Повна перевірка системи на відповідність технічним специфікаціям.

Операційне тестування: Перевірка готовності програми до випуску.

Приймальне тестування: Оцінка програми кінцевими користувачами на предмет задоволення їх потреб.

Тест-план для ручного тестування описує повний обсяг роботи тестувальника, включаючи цілі, ресурси, графіки, стратегії, критерії початку і завершення тестування, а також обладнання і методи проєктування тестів. Також важливо оцінити всі можливі ризики тестування і розробити стратегії для їх вирішення.

Переваги ручного тестування включають:

Користувацький feedback: Можливість отримати детальний звіт від тестувальника, як від повноцінного користувача.

Інтерфейсний feedback: Повне тестування користувацького інтерфейсу.

Економічна вигода: Зменшення витрат у порівнянні з автоматизованими методами на певних етапах розробки.

Тестування в реальному часі: Можливість тестування на ранніх етапах розробки.

Можливість дослідницького тестування: Гнучкість у виборі об'єктів для тестування.

Недоліки ручного тестування:

Людський фактор: Підвищений ризик людських помилок.

Неможливість навантажувального тестування: Відсутність засобів для перевірки під високим навантаженням.

Складність повторної перевірки: Важкість в ідентифікації помилок після внесення змін у код.

Автоматизоване тестування забезпечує більшу швидкість і точність у виконанні повторюваних тестів за допомогою програмних інструментів. Однак воно також вимагає значних початкових інвестицій в розробку тестових сценаріїв і покупку інструментів автоматизації.

Типи автоматизованого тестування

Модульне тестування: Тестування окремих компонентів програми для перевірки їх ізоляційної працездатності.

Інтеграційне тестування: Тестування комбінацій модулів та взаємодій між ними.

Системне тестування: Перевірка комплексної системи як єдиного цілого.

Регресивне тестування: Перевірка, що попередньо робочий функціонал залишився не зміненим після оновлень чи внесення нових функцій.

Навантажувальне тестування: Тестування системи на стійкість і продуктивність при різних умовах навантаження.

Переваги автоматизованого тестування включають:

Швидкість виконання: Автоматизація дозволяє виконувати тести значно швидше, ніж ручне тестування.

Повторюваність: Тестові сценарії можна запускати скільки завгодно разів, з мінімальними зусиллями щодо повторення.

Точність: Зменшується ймовірність помилок, які може допустити людина.

Краще використання ресурсів: Автоматизація дозволяє звільнити час

тестувальників для виконання більш складних задач.

Масштабування: Тести можуть бути легко масштабовані для перевірки системи під високим навантаженням або великою кількістю користувачів.

Недоліки автоматизованого тестування включають:

Висока вартість впровадження: Розробка та підтримка тестових скриптів може бути дорогою.

Складність налаштувань: Для ефективного автоматизованого тестування потрібні висококваліфіковані спеціалісти.

Обмеження за обсягом: Не всі види тестування можна автоматизувати, особливо ті, що вимагають суб'єктивної оцінки, наприклад, тести користувацького інтерфейсу або досвіду користувача.[2]

1.3. Класифікація тестування

Тестування програмного забезпечення може бути класифіковано за багатьма критеріями, що включають типи тестів, мету та методологію виконання. Розуміння цих класифікацій допомагає тестувальникам обрати відповідний підхід та інструментарій для ефективної перевірки програмних продуктів.[4]

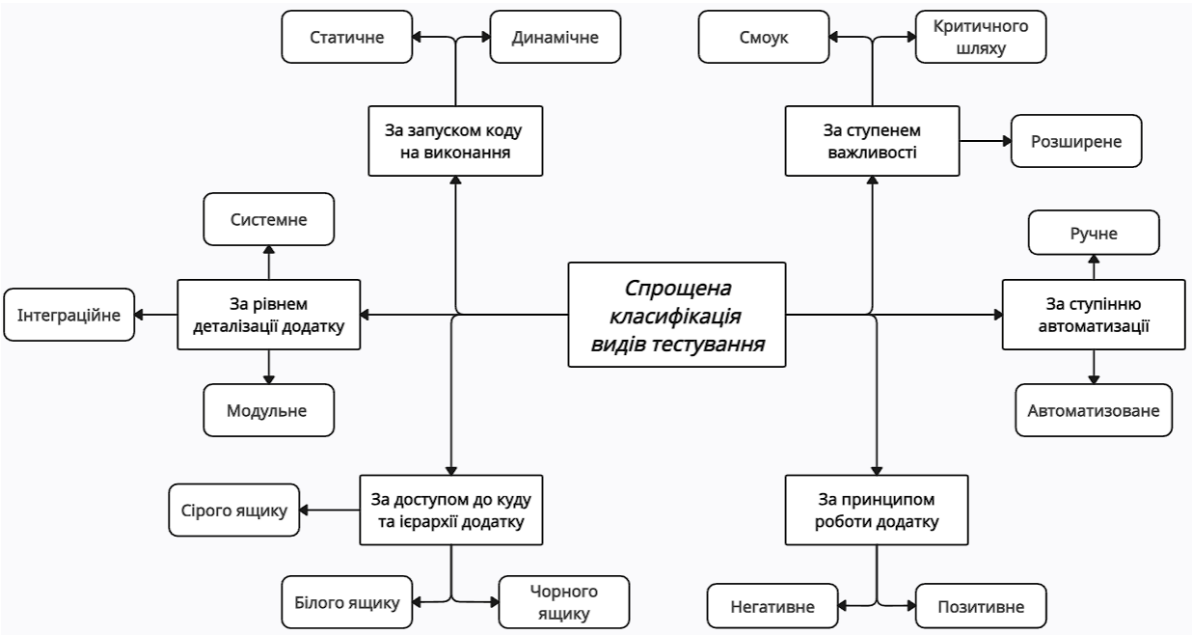


Рисунок 1.1 - Спрощена класифікація тестування.

1. Тестування за запуском коду на виконання.

Статичне тестування – тип тестування, який припускає, що програмний код під час тестування не буде виконуватися. При цьому саме тестування може бути як ручним, так і автоматизованим.

Статичне тестування починається на ранніх етапах життєвого циклу ПЗ і є, відповідно, частиною процесу верифікації. Для цього типу тестування в

деяких випадках навіть не потрібен комп'ютер, – наприклад, при перевірці вимог.

Статичне тестування також може бути автоматизоване – наприклад, можна використовувати автоматичні засоби перевірки синтаксису програмного коду.

Динамічне тестування – тип тестування, який передбачає запуск програмного коду. Таким чином, аналізується поведінка програми під час її роботи.

Для виконання динамічного тестування необхідно, щоб програмний код, що тестується, був написаний, скомпільований та запущений. При цьому, може виконуватися перевірка зовнішніх параметрів роботи програми: завантаження процесора, використання пам'яті, час відповіді та т.д. – тобто, її продуктивність.[3]

2. За ступенем автоматизації.

Ручне тестування

Автоматизоване тестування

3. За рівнем деталізації додатку.

Модульне тестування (Unit Testing) – Фокусується на перевірці окремих модулів або компонентів ПЗ. Зазвичай проводиться розробниками для забезпечення коректності базових функцій.

Інтеграційне тестування (Integration Testing) – Перевіряє взаємодію між різними модулями чи компонентами системи. Виявляє помилки в інтерфейсах та взаємодіях між складовими частинами ПЗ.

Системне тестування (System Testing) – Оцінює повністю інтегровану систему щодо її відповідності до зазначених вимог. Виконується на зібраному продукті, який вважається готовим до випуску.

4. За доступом до коду та ієрархії програми.

Чорний ящик (Black Box Testing) – Тестування без знання внутрішньої структури системи. Тестувальник зосереджується на вхідних даних та вихідних результатах, не аналізуючи внутрішні процеси програми.

Білий ящик (White Box Testing) – Вимагає знань внутрішньої структури та коду програми. Тестування може включати аналіз потоків даних, логічних умов, циклів і інших аспектів внутрішньої реалізації.

Сірий ящик (Grey Box Testing) – Комбінація методів чорного та білого ящика. Тестувальники мають частковий доступ до внутрішньої структури системи, що допомагає оптимізувати тестування та зосередитись на критичних аспектах.

5. За ступенем важливості тестових функцій.

Смоук-тестування(димове) – це мінімальний набір тестів, який перевіряє основні функції системи або компонента. Завдання – перевірити, чи працює система в загальних рисах, без детальної перевірки.

Тестування критичного шляху – Перевіряються найбільш важливі елементи і функції програми, правильність роботи та їх використання.

Розширене тестування – Перевірка загальної функціональності.

6. За принципом роботи з додатком

Позитивне тестування (Positive Testing) – (також відоме як "тестування за номіналом"), зосереджене на тому, щоб переконатися, що програма правильно виконується на валідних вхідних даних. Цей підхід припускає, що користувач використовує систему так, як передбачено, і всі вхідні дані є коректними.

Негативне тестування (Negative Testing) – використовується для перевірки, як програмне забезпечення реагує на неправильні, неочікувані або

крайні вхідні дані. Цей тип тестування дуже важливий, оскільки допомагає ідентифікувати потенційні вразливості або помилки, які можуть виникнути внаслідок неправильного використання продукту.

1.4. Автоматизація тестування

Автоматизація тестів є найбільш діючим способом підвищення ефективності тестування та зменшення часу тестування ПЗ.[7]

Автоматизоване тестування програмного забезпечення є важливим з кількох причин:

1. Ручне тестування кожного робочого процесу, поля і негативного сценарію може бути дорогим і часомістким.

2. Вручну складно перевіряти сайти, що підтримують декілька мов.

3. Автоматизація дозволяє проводити тести без постійного нагляду людини, що економить час і ресурси.

4. Автоматизація значно прискорює процес тестування.

5. Автоматизовані тести дозволяють значно розширити охоплення тестами.

6. Ручне тестування може стати монотонним і схильним до помилок через людський фактор.

Під час автоматизованого тестування виконуються такі кроки:

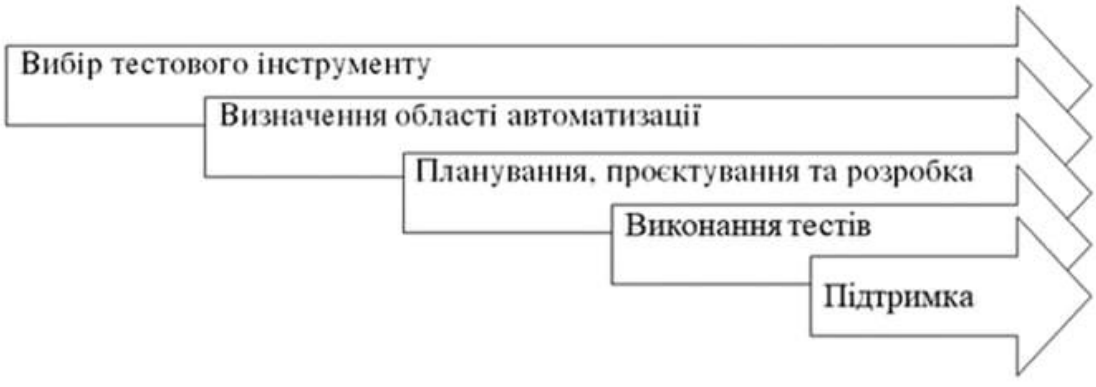


Рисунок 1.2 - Процеси автоматизованого тестування.

З зазначених п’яти кроків найбільш важливими є перші два, оскільки від визначення того що ми можемо протестувати автоматично, а також в залежності від вибору тестового інструменту буде залежати абсолютно всі подальші кроки.

1.5. Поняття *Web* – додатку

Веб-додаток — це ПЗ, що працює в Інтернеті та доступне за допомогою веб-браузера. Складається з двох частин: клієнтської частини, яка завантажується в браузер користувача, і серверної частини, яка обробляє запити користувача і повертає результати.

Веб-додаток — це клієнт-серверний додаток, в якому взаємодія між клієнтом і веб-сервером здійснюється через веб-браузер. Програмне забезпечення веб-додатків включає програми та дані, розроблені для глобального використання та взаємодії, поєднуючи можливості мережевих і клієнт-серверних технологій та обладнання для обслуговування як локальних, так і віддалених користувачів.

Для ефективного тестування веб-програмного забезпечення, як і будь-якого іншого типу програмного забезпечення, тестувальникам необхідно глибоко розуміти особливості такого програмного забезпечення, його поведінку та взаємодію з іншими програмними і апаратними компонентами.

Як і в інших типах програмного забезпечення, у веб-додатків є свої особливості та потенційні ризики, які можуть призвести до збоїв у програмі або системі. Виявлення та усунення таких проблемних місць є ключовими завданнями під час процесу тестування.[12]

Клієнт-серверна архітектура базується на двох основних елементах: клієнті та сервері.

Клієнт – це комп'ютер, розташований на боці користувача, який ініціює запити до сервера з метою отримання інформації чи виконання специфічних операцій.

Сервер – це потужніший комп'ютер або спеціалізоване обладнання, призначене для обробки програмного коду, виконання задач за запитами клієнтів, забезпечення доступу до ресурсів, а також зберігання даних та баз даних.

У такій системі клієнт відправляє запити на сервер, де вони обробляються, і результати повертаються назад клієнту. Сервер має здатність

обслуговувати численних клієнтів одночасно. У випадку надходження декількох запитів одночасно, вони організовуються у чергу і обробляються по черзі. Часом, деякі запити можуть мати вищий пріоритет, що означає їхнє пріоритетне виконання.[15]

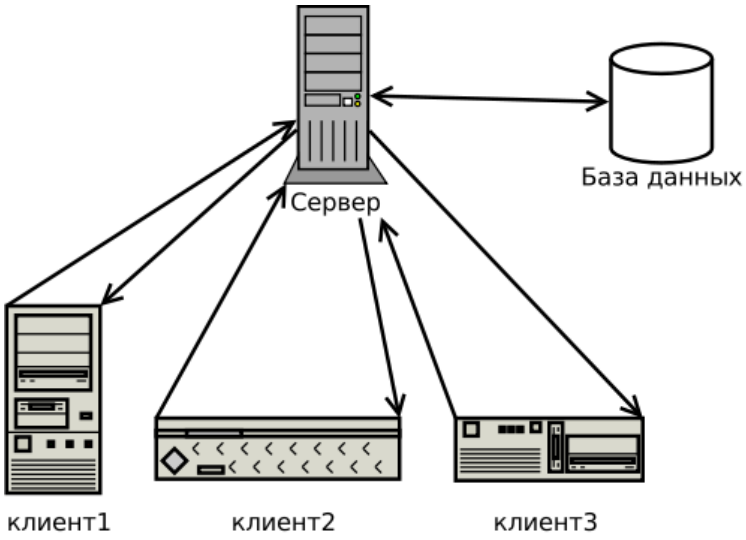


Рисунок 1.3 - Клієнт-серверна архітектура.

Функції, які реалізуються на сервері:

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

Функції, які реалізуються на стороні клієнта:

1. надання користувальницького інтерфейсу;
2. формулювання запиту до сервера і його відправка;
3. отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

```
GET /path HTTP/1.0
Content-Type: text/html; charset=utf-8
Content-Length: 4
X-Custom-Header: value

test
```

Рисунок 1.4 - Приклад *Http* запиту.

У першому рядку вказано метод запиту - GET, шлях до ресурсу - / path і версія протоколу - HTTP / 1.0.

Далі йде блок заголовків. Заголовки - це пари ключ: значення, кожна з яких записується з нового рядка і поділяється двокрапкою. Вони передають додаткові дані та налаштування від клієнта до сервера та назад.

HTTP — це текстовий протокол, тому всі дані передаються у вигляді тексту. Заголовки можна відокремити один від одного лише перенесенням рядка. Не можна використовувати коми, крапку з комою або інші роздільники. Все, що йде після імені заголовка з двокрапкою і до перенесення рядка буде вважатися значенням заголовка.

```
HTTP/1.1 200 OK
Date: Thu, 29 Jul 2021 19:20:01 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2
Connection: close
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

OK
```

Рисунок 1.5 - Приклад відповіді сервера.

Статуси відповідей

Клієнту часто недостатньо просто надіслати запит на сервер. У багатьох випадках треба дочекатися відповіді та зрозуміти, як сервер обробив запит. Для цього було придумано статуси відповідей. Це трицифрові числові коди з невеликими текстовими позначеннями. Їх можна побачити у терміналі чи браузері. Самі коди поділяються на 5 класів:[13]

Інформаційні відповіді: коди 100-199

Успішні відповіді: коди 200–299

Редиректи: коди 300-399

Клієнтські помилки: коди 400-499

Серверні помилки: коди 500–599

Висновок

У даному розділі було розглянуті основи тестування ПЗ. Було розкрите поняття тестування , його типи та види. Проаналізовано спрощену класифікацію тестування. Розглянуті основні відмінності ручного та автоматизованого тестування, їх плюси та мінуси. Розібрані основні техніки тестування ПЗ. Розібране поняття Web-додатку та його особливості. Охарактеризовано структуру архітектури клієнт-серверного додатку , розглянуто вигляд запитів та відповідей. Розписано основні типи відповідей сервера та їх значення.

2 РОЗДІЛ СУЧАНІ ІНСТУМЕНТИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ WEB-ДОДАТКІВ

Автоматизоване тестування веб-додатків стало невід'ємною частиною сучасного процесу розробки програмного забезпечення. З постійним зростанням складності веб-додатків і вимог до їхньої якості, автоматизація тестування дозволяє значно підвищити ефективність процесу тестування, забезпечуючи більш швидке виявлення та виправлення помилок. Сучасні інструменти автоматизованого тестування пропонують широкий спектр можливостей для створення, виконання та аналізу тестів, що дозволяє розробникам і тестувальникам фокусуватися на покращенні функціональності та стабільності своїх продуктів. У цьому розділі буде розглянуто основні інструменти автоматизованого тестування веб-додатків, їхні особливості, переваги та недоліки, що допоможе вибрати найбільш підходящий інструмент для нашого проекту.

2.1. Топ інструментів станом на 2024

Засоби автоматизованого тестування (інструменти автоматизованого тестування) – це програмні рішення, які використовуються для автоматизації процесу тестування програмного забезпечення. Вони дозволяють розробникам і тестувальникам створювати, виконувати та аналізувати тестові сценарії без необхідності вручну виконувати кожен тест. Ці інструменти допомагають підвищити ефективність тестування, зменшити кількість помилок і прискорити випуск якісного програмного забезпечення.

Наведемо найпопулярніші на даний момент засоби автоматизації тестування:

1. *Selenium*
2. *Katalon Studio*
3. *Cypress*
4. *Testim*

Дані 4 засоби є автоматизації тестування є одними з найкращих на даний момент.

Для більшого розуміння преваг та недоліків кожного інструменту зробимо порівняльну таблицю в яку ввійдуть основні особливості та характеристики кожного з них (таблиця 2.1).

Таблиця. 2.1 Порівняльна характеристика.

Інструмент	*Katalon Studio*	*Selenium*	*Testim*	*Cypress*
Види тестування	Функціональне, регресійне, API, мобільне	Функціональне, регресійне, перформанс, API	Функціональне, регресійне, енд-ту-енд	Функціональне, регресійне, енд-ту-енд, компонентне
Підтримувані ОС	Windows, macOS	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux
Підтримувані мови	Groovy, Java, JavaScript	ava, C#, Python, Ruby, JavaScript	JavaScript, TypeScript	JavaScript, TypeScript
Наявність ШІ	Так, використовує ШІ для адаптивного тестування	Ні	Так, використовує ШІ для покращення стабільності тестів	Ні
Легкість використання	Висока, інтуїтивний інтерфейс	Низька, потребує знань програмування та налаштувань	Висока, інтуїтивний інтерфейс з підказками	Середня, потребує знань JavaScript і налаштувань

Продовження таблиці. 2.1 Порівняльна характеристика.

Доступність(ціна)	Безкоштовна версія, платні розширення	Безкоштовний	Платний, є пробний період	Безкоштовний

2.2. Детальна характеристика *Selenium*

Selenium є одним із найпопулярніших і найпотужніших інструментів для автоматизованого тестування веб-додатків. Його гнучкість і широка підтримка роблять його вибором багатьох організацій та індивідуальних розробників. Ось детальна характеристика Selenium:

Історія та розвиток

Selenium був розроблений Джейсоном Хаггінсом у 2004 році як внутрішній інструмент для тестування в ThoughtWorks. З часом інструмент набув популярності та отримав активну підтримку з боку спільноти розробників.

Архітектура

Selenium складається з декількох компонентів:

1. Selenium WebDriver: Основний компонент, який дозволяє автоматизувати взаємодію з браузерами. Він забезпечує програмний інтерфейс для взаємодії з веб-додатками.

2. Selenium IDE: Інтегрована середа розробки для запису та відтворення тестів без програмування. Ідеально підходить для швидкого створення тестів.

3. Selenium Grid: Дозволяє розподіляти виконання тестів на декількох машинах та в різних середовищах паралельно, що прискорює процес тестування.

4. Selenium RC (Remote Control): Раніше був основним інструментом для автоматизації, але з часом його замінив Selenium WebDriver.

Підтримувані мови програмування

Selenium підтримує широкий спектр мов програмування, включаючи:

- *Java*

- *C#*

- *Python*

- *Ruby*

- *JavaScript*

- *PHP*

Це робить Selenium універсальним інструментом, який може бути інтегрований у різні середовища розробки.

Підтримувані браузери та операційні системи

Selenium WebDriver працює з багатьма популярними браузерами, включаючи:

- *Google Chrome*

- *Mozilla Firefox*

- *Internet Explorer*

- *Microsoft Edge*

- *Safari*

- *Opera*

Підтримувані операційні системи:

- *Windows*

- *macOS*

- *Linux*

Особливості Selenium

Крос-браузерне тестування: Selenium дозволяє тестувати веб-додатки на різних браузерах і платформах, забезпечуючи їхню сумісність та функціональність.

Гнучкість: Завдяки підтримці різних мов програмування та широкій інтеграції, Selenium легко інтегрується в існуючі процеси розробки та CI/CD.

Масштабованість: Використовуючи Selenium Grid, можна розподіляти тестові навантаження на декілька машин, що значно скорочує час виконання тестів.

Відкрите джерело: Selenium є інструментом з відкритим вихідним

кодом, що дозволяє спільноті розробників вносити свій вклад у його розвиток
та адаптувати під свої потреби.

Широка спільнота: Велика спільнота користувачів та розробників забезпечує постійну підтримку, оновлення та безліч ресурсів для навчання та вирішення проблем.

Види тестування

Selenium використовується для різних видів тестування:

Функціональне тестування: Перевірка функціональності веб-додатків.

Регресійне тестування: Перевірка того, що нові зміни не порушили існуючу функціональність.

Перформанс тестування: Хоча Selenium не спеціалізується на тестуванні продуктивності, його можна інтегрувати з іншими інструментами для цього виду тестування.

Тестування API: За допомогою сторонніх бібліотек та фреймворків можна тестувати веб-сервіси та API.

Тестування користувацького інтерфейсу (UI): Перевірка роботи інтерфейсу користувача, включаючи елементи управління та відображення контенту.

Виклики та обмеження

Складність налаштування: Налаштування Selenium може бути складним, особливо для новачків. Вимагає знань програмування та розуміння браузерних драйверів.

Підтримка динамічного контенту: Selenium може мати проблеми з тестуванням динамічно змінюваного контенту, що вимагає додаткових налаштувань і синхронізації.

Швидкість виконання тестів: Виконання тестів через інтерфейс користувача може бути повільним порівняно з іншими методами автоматизації.

Масштабованість: Незважаючи на можливість використання Selenium

Grid, налаштування великомасштабного тестування може вимагати значних

зусиль і ресурсів.

Інтеграція та розширення[9]

Selenium можна інтегрувати з багатьма іншими інструментами та фреймворками, включаючи:

JUnit, TestNG: Інструменти для організації та виконання тестів на Java.

Maven, Gradle: Системи управління збіркою для автоматизації процесу тестування.

Jenkins, Bamboo: Інструменти CI/CD для автоматизації тестування та розгортання.

Appium: Інструмент для автоматизованого тестування мобільних додатків, який базується на Selenium WebDriver.

Приклад використання

java Копировать код

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class SeleniumExample {
    public static void main(String[] args) {
        // Вказуємо шлях до драйвера Chrome
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        // Створюємо екземпляр драйвера
        WebDriver driver = new ChromeDriver();

        // Відкриваємо веб-сторінку
        driver.get("https://www.example.com");

        // Знаходимо елемент на сторінці
        WebElement element = driver.findElement(By.name("q"));

        // Вводимо текст у поле пошуку
        element.sendKeys("Selenium WebDriver");

        // Відправляємо форму
        element.submit();

        // Закриваємо браузер
        driver.quit();
    }
}
```

Рисунок 2.1 - Тестовий сценарій на Java з використанням Selenium WebDriver:

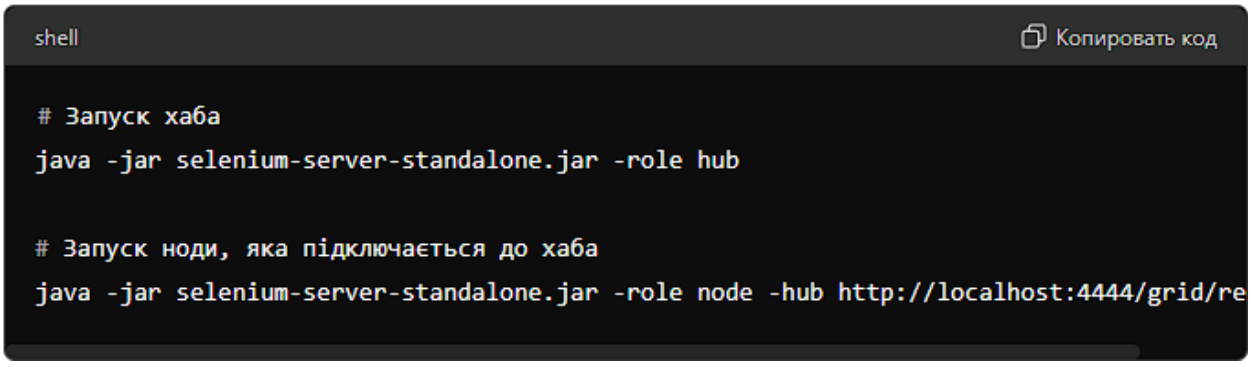


Рисунок 2.2 - Конфігурація Selenium Grid для паралельного виконання тестів

Selenium є потужним та гнучким інструментом для автоматизації тестування веб-додатків. Його широка підтримка мов програмування, браузерів та операційних систем робить його універсальним рішенням для різних потреб. Незважаючи на певні виклики та обмеження, активна спільнота та постійний розвиток роблять Selenium незамінним інструментом для автоматизованого тестування.

2.3. Детальна характеристика *Katalon Studio*

Katalon Studio — це потужний інструмент для автоматизації тестування, який дозволяє тестувати веб-додатки, мобільні додатки, API та настільні додатки. Відзначається своєю простотою у використанні, багатофункціональністю та підтримкою різних середовищ тестування.

Історія та розвиток

Katalon Studio був випущений у 2015 році компанією Katalon LLC. Інструмент розроблений на базі фреймворків Selenium та Appium, забезпечуючи користувачам простий і ефективний спосіб автоматизувати тестування.

Архітектура

Katalon Studio має зручний графічний інтерфейс та вбудовані функції для створення, виконання та керування тестами. Основні компоненти включають:

1. Katalon Recorder: Розширення для браузерів Chrome та Firefox для запису та відтворення тестів.

2. Katalon Studio IDE: Інтегроване середовище розробки для створення та управління тестовими сценаріями.

3. Katalon TestOps: Платформа для управління тестами, аналізу результатів та підвищення ефективності тестування.

Підтримувані мови програмування

Katalon Studio використовує Groovy, мову сценаріїв, яка базується на Java. Це дозволяє користувачам легко писати складні тестові сценарії та використовувати функціональність Java.

Підтримувані браузери та операційні системи

Katalon Studio підтримує автоматизацію тестування на різних браузерах, включаючи:

- *Google Chrome*
- *Mozilla Firefox*
- *Internet Explorer*
- *Microsoft Edge*
- *Safari*

Підтримувані операційні системи:

- *Windows*
- *macOS*
- *Linux*

Особливості Katalon Studio

Крос-платформна підтримка: Підтримує тестування веб-додатків, мобільних додатків, API та настільних додатків.

Інтуїтивно зрозумілий інтерфейс: Зручний графічний інтерфейс дозволяє швидко створювати та виконувати тести без необхідності глибоких знань програмування.

Інтеграція з CI/CD інструментами: Підтримує інтеграцію з Jenkins, Bamboo, Azure DevOps та іншими інструментами CI/CD.

Гнучкість: Дозволяє створювати тести за допомогою запису та відтворення або вручну писати скрипти для складніших сценаріїв.

Вбудовані функції звітності: Генерує детальні звіти про виконання тестів, включаючи логи, скріншоти та іншу інформацію.

Підтримка тестування API: Забезпечує автоматизацію тестування RESTful та SOAP API з можливістю перевірки відповідей та поведінки. Види тестування

Katalon Studio використовується для різних видів тестування:

Функціональне тестування: Перевірка функціональності веб-додатків, мобільних додатків та API.

Регресійне тестування: Перевірка того, що нові зміни не порушили існуючу функціональність.

Тестування продуктивності: Можливість інтеграції з JMeter для тестування продуктивності.

Тестування користувацького інтерфейсу (UI): Перевірка роботи інтерфейсу користувача, включаючи елементи управління та відображення контенту.

Тестування мобільних додатків: Підтримує автоматизацію тестування на Android та iOS платформах.

Тестування API: Автоматизація тестування веб-сервісів, перевірка відповідей та поведінки API.

Виклики та обмеження

Навчання та адаптація: Хоча інструмент інтуїтивно зрозумілий, новачкам може знадобитися час на ознайомлення з його можливостями та налаштуваннями.[8]

Обмежена підтримка мов програмування: Використання лише Groovy може бути обмеженням для деяких команд, які вважають за краще використовувати інші мови програмування.

Вартість: Повнофункціональна версія Katalon Studio є комерційним продуктом, що може бути обмежуючим фактором для невеликих компаній або окремих розробників.

Інтеграція та розширення

Katalon Studio підтримує інтеграцію з багатьма інструментами та платформами, включаючи:

Jenkins, Bamboo, Azure DevOps: Інструменти CI/CD для автоматизації тестування та розгортання.

JIRA, qTest, TestRail: Інструменти для управління тестуванням та дефектами.

Slack, Microsoft Teams: Інтеграція для повідомлень про результати тестування.

JMeter: Інтеграція для тестування продуктивності.

Приклад виикористання

groovy Копировать код

```groovy
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI

// Відкриваємо браузер
WebUI.openBrowser('')

// Переходимо на веб-сайт
WebUI.navigateToUrl('https://www.example.com')

// Вводимо текст у поле пошуку
WebUI.setText(findTestObject('Page_Example/input_Search_q'), 'Katalon Studio')

// Натискаємо кнопку пошуку
WebUI.click(findTestObject('Page_Example/button_Search'))

// Закриваємо браузер
WebUI.closeBrowser()
```

Рисунок 2.3 - Створення тестового сценарію для веб-додатку

groovy Копировать код

```groovy
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS

// Відправляємо GET запит
def response = WS.sendRequest(findTestObject('API/GetUser'))

// Перевіряємо статус-код
WS.verifyResponseStatusCode(response, 200)

// Перевіряємо значення у відповіді
WS.verifyElementPropertyValue(response, 'username', 'testuser')
```

Рисунок 2.4 - Автоматизація тестування API

Katalon Studio — це потужний інструмент для автоматизації тестування, який забезпечує широкий спектр можливостей для тестування веб-додатків, мобільних додатків, API та настільних додатків. Його інтуїтивно зрозумілий інтерфейс, підтримка різних видів тестування та інтеграція з популярними інструментами роблять його універсальним рішенням для автоматизованого тестування. Незважаючи на певні обмеження, такі як обмежена підтримка мов програмування та вартість, Katalon Studio є відмінним вибором для команд, які шукають комплексне рішення для автоматизації тестування.

2.4. Детальна характеристика *Cypress*

Cypress — це сучасний інструмент для автоматизації тестування, орієнтований на тестування веб-додатків. Він відзначається простотою використання, швидкістю виконання тестів і інтеграцією з сучасними інструментами розробки.

Історія та розвиток

Cypress був створений у 2014 році і з тих пір отримав визнання серед розробників і тестувальників завдяки своїй простоті та потужності. Основний фокус інструменту — забезпечення швидкого та зручного автоматизованого тестування фронтенд-розробок.

Архітектура

Cypress побудований з урахуванням потреб розробників, надаючи потужний API для написання тестів і можливість виконання тестів безпосередньо в браузері. Архітектура включає:

1. Cypress Test Runner: Інтерфейс для запуску і перегляду тестів у реальному часі.

2. Cypress Dashboard: Платформа для управління тестовими запусками, аналізу результатів і відстеження історії тестувань.

3. Cypress CLI: Командний інтерфейс для інтеграції Cypress з CI/CD процесами.

Підтримувані мови програмування

Cypress використовує JavaScript, що робить його популярним серед фронтенд-розробників, які вже знайомі з цією мовою. Сценарії пишуться за допомогою Mocha та Chai, популярних бібліотек для тестування в JavaScript.

Підтримувані браузери та операційні системи[10]

Cypress підтримує автоматизацію тестування на наступних браузерах:

- *Google Chrome*
- *Mozilla Firefox*
- *Microsoft Edge*

Підтримувані операційні системи:

- *Windows*
- *macOS*
- *Linux*

Особливості Cypress

Реальне середовище виконання: Тести виконуються безпосередньо в браузері, що дозволяє бачити результати в реальному часі.

Швидкість: Cypress швидко виконує тести завдяки своїй архітектурі та інтеграції з браузером.

Легка інтеграція: Легко інтегрується з іншими інструментами CI/CD, такими як Jenkins, CircleCI, Travis CI та інші.

Детальна документація та велика спільнота: Наявність детальної документації та активної спільноти розробників полегшує вивчення та використання інструменту.

Зручний API: Cypress надає простий та зрозумілий API для написання тестів, що робить його доступним навіть для початківців.

Автоматичне очікування: Вбудована підтримка автоматичного очікування завершення запитів та завантаження елементів, що зменшує необхідність використання команд типу wait.

Види тестування

Cypress підтримує різні види тестування:

Енд-то-енд тестування (E2E): Перевірка всієї функціональності

додатка з точки зору користувача.

Інтеграційне тестування: Тестування окремих компонентів і їх взаємодії між собою.

Тестування компонентів: Перевірка окремих компонентів додатка на рівні UI.

Виклики та обмеження

Підтримка браузерів: Обмежена кількість підтримуваних браузерів порівняно з іншими інструментами автоматизації.

Складність налаштування для деяких сценаріїв: Хоча Cypress є зручним для більшості випадків використання, складніші сценарії можуть вимагати додаткових налаштувань.

Проблеми з тестуванням мобільних додатків: Cypress орієнтований переважно на тестування веб-додатків і не підтримує мобільні платформи.

Інтеграція та розширення

Cypress інтегрується з багатьма інструментами та платформами:

CI/CD інструменти: Jenkins, CircleCI, Travis CI, GitLab CI/CD та інші.

Сповіщення: Slack, Microsoft Teams та інші.

Інструменти управління тестами: TestRail, QTest та інші.

Приклади використання

javascript Копировать код

```javascript
describe('My First Test', () => {
  it('Visits the Kitchen Sink', () => {
    cy.visit('https://example.cypress.io')
    cy.contains('type').click()
    cy.url().should('include', '/commands/actions')
    cy.get('.action-email')
      .type('fake@email.com')
      .should('have.value', 'fake@email.com')
  })
})
```

Рисунок 2.5 - Створення простого енд-то-енд тесту

```javascript
import { mount } from '@cypress/react'
import MyComponent from './MyComponent'

describe('MyComponent', () => {
  it('renders correctly', () => {
    mount(<MyComponent />)
    cy.get('h1').should('contain', 'Hello, World!')
  })
})
```

Рисунок 2.6 - Тестування компоненту

Cypress — це потужний інструмент для автоматизації тестування веб-додатків, який забезпечує високу швидкість виконання тестів, легкість використання та інтеграцію з сучасними інструментами розробки. Його архітектура, орієнтована на реальне середовище виконання, робить його ідеальним вибором для тестування фронтенд-розробок. Незважаючи на деякі обмеження, такі як обмежена підтримка браузерів та відсутність підтримки мобільних платформ, Cypress залишається одним з найпопулярніших інструментів для автоматизації тестування завдяки своїй простоті та потужності.

2.5. Детальна характеристика *Testim*

Testim - це інструмент для автоматизації тестування, що спрощує процес створення, управління та виконання тестів для веб-додатків. Він використовує штучний інтелект для створення стабільних та надійних тестів, що допомагає розробникам та тестувальникам швидше виявляти та виправляти помилки.

Історія та розвиток

Testim був створений у 2015 році і з тих пір став досить популярним інструментом в області автоматизації тестування веб-додатків. Він пропонує інтуїтивний інтерфейс та широкі можливості для створення складних тестів.

Архітектура

Testim використовує хмарну архітектуру, що дозволяє виконувати тести на різних конфігураціях та пристроях без зайвих обмежень. Він має такі ключові компоненти:

1. Testim Editor: Інтерфейс для створення та редагування тестів за допомогою простих дій перетягування та випадаючих списків.

2. Testim CLI: Командний інтерфейс для інтеграції Testim з CI/CD системами та автоматизацією тестування.

3. Testim Dashboard: Інтерфейс для управління тестами, перегляду результатів та аналізу виконання тестів.

Підтримувані мови програмування

Testim підтримує автоматизацію тестів з використанням JavaScript, що робить його зручним для великої кількості розробників, що працюють з веб-додатками.[11]

Підтримувані браузери та операційні системи

Testim підтримує автоматизацію тестів у різних браузерах, таких як Google Chrome, Mozilla Firefox, Microsoft Edge, Safari тощо. Щодо операційних систем, він підтримує Windows, macOS, Linux.

Особливості Testim

Сучасний інтерфейс редагування тестів: Інтуїтивний інтерфейс робить процес створення тестів швидким та простим.

Штучний інтелект для стабільних тестів: Testim використовує AI для виявлення та усунення проблем у тестах.

Широкі можливості інтеграції: Інтеграція зі звичайними CI/CD системами, такими як Jenkins, CircleCI, GitHub Actions тощо.

Вбудований рекордер тестів: Можливість записувати дії тестувальника та перетворювати їх на автоматичні тести.

Види тестування

Testim підтримує різні види тестування, включаючи енд-то-енд тестування, тестування інтерфейсу користувача, тестування компонентів та інші.

Виклики та обмеження

Необхідність у вивченні нового інструменту: Хоча Testim має інтуїтивний інтерфейс, він все одно вимагає часу для того, щоб зрозуміти всі його можливості.

Обмежена можливість редагування тестів: Деякі складні тестові сценарії можуть вимагати більшої гнучкості у редагуванні, ніж це може забезпечити Testim.

Інтеграція та розширення

Testim може бути інтегрований з багатьма іншими інструментами та платформами, такими як Slack, Jira, GitHub, Jenkins, CircleCI та інші.

Приклад використання

Рисунок 2.5 - Створення енд-то-енд тесту

Testim - це потужний інструмент для автоматизації тестування веб-додатків, який використовує штучний інтелект для створення стабільних та надійних тестів. Він має інтуїтивний інтерфейс, що дозволяє швидко створювати та редагувати тести, а також широкі можливості інтеграції з іншими інструментами та CI/CD системами. Testim підтримує різні види тестування та операційні системи та браузери, що робить його відмінним вибором для тестування веб-додатків у різних середовищах.

Висновок

На даний момент існує величезна кількість засобів автоматизованого тестування. Кожен інструмент має свої переваги та недоліки. Такий великий вибір засобів значно ускладнює вибір підходящого для нашого проєкту. Нами було розглянуті найбільш популярні та цікаві інструменти для автоматизованого тестування web додатків, були розкриті їх плюси та мінуси, розглянуті найголовніші та найцікавіші особливості кожного з них. Уся розглянута нами інформація допоможе обрати найбільш підходящий для нашого проєкту інструмент.

3 РОЗДІЛ ТЕСТУВАННЯ WEB-ДОДАТКУ

3.1. Опис об’єкта тестування

Необхідно описати web-додаток для якого буде розроблено тест-план та проведене ручне та автоматизоване тестування.

Для автоматизованого тестування оберемо сайт інтернет магазину для тестування звичного більшості функціоналу, на кшталт web-додатку магазину електронної техніки: *https://allo.ua/ua/*

Allo – сайт, що містить усі функції стандартного інтернет магазину.

Під час використання сайту можна переглядати каталог товарів, дивитися їх характеристики та порівнювати між собою, увійти до акаунту, додавати товари до кошика, оформляти заявку, зв’язатися з підтримкою.

В процесі тестування сайту *Allo* будуть розроблені тест кейси за допомогою яких буде зроблена перевірка усіх основних функцій сайту, що доступні користувачу.

Процес тестування розділимо на 4 етапи:

1)Складання тест плану та розробка функціональних тестів;

2)Проведемо ручне тестування сайту;

3)Проведемо автоматизоване тестування у вибраному середовищі;

4)Порівняємо отримані результати ручного та автоматизованого тестування;

5)Написання загального висновку по результатам тестування готового продукту.

3.2. Вибір інструмента для автоматичного тестування

У другому розділі нами були розглянуті 4 найпопулярніших та найцікавіших інструменти автоматичного тестування.[8]

Для автотестування нашого сайту оберемо *Katalon Studio*, як інструмент перевірений часом та актуальними на даний момент функціями перевірки з використанням ШІ.

На даний момент *Katalon Studio* є найкращім інструментом автоматизації тестування з боку кількості представлених у ньому функцій. *Katalon Studio,* як і будь який сучасний засіб для автотестів, надає можливість імітувати рух миші та натискання клавіш на клавіатурі від імені користувача.

Katalon Studio є ультимативним інструментом автоматизації браузера, через це його використання та налаштування не є занадто складним.

Для початку роботи нам потрібно встановити сам додаток від *Katalon Studio.*

Це можна зробити на офіційному сайті.

https://katalon.com/

Після завантаження ПЗ на потрібно зареєструватися для отримання пробного періоду, що складає 30 днів. Далі на нас чекає невелике навчання базовим функціям програми, що допоможе початківцю у сфері тестування розібратися з потенціалом *Katalon Studio* як сервіса для автоматизації тестування.

У *Katalon Studio* є можливість безпосередньої роботи з відкритим кодом для написання власних скриптів так і вбудований рекордер для автоматичного запису дій користувача та перетворення їх на автоматизовані тести.

Katalon Studio підтримує такі мови програмування як Groovy та Java також є можливість інтеграції з Jenkins, Git, JIRA та іншими інструментами для безперервної інтеграції та розгортання.

3.3. Процес документації

Після встановлення ПЗ можно перейти до оформлення документації нашого тестування.

Згідно з першим розділом нашого плану тестування ми маємо скласти тест план та розробити функціональні тести для нашого сайту.

До нашого тест плану увійдуть наступні пункти:

1. Вступ.
2. Мета.
3. Вихідні дані до проєкту.
4. Мета тестування.
5. Умови тестування.
6. Функціональне тестування.
7. Тестування кросбраузерності.
8. Візуальне тестування.
9. Результат тестування.
10. Висновки;

Мета тест плану: цілью нашого тест плану є опис процесу тестування сайту *Allo* (https://allo.ua/ua/). Даний опис дозволить більш поглиблено зрозуміти процес роботи QA інженера.

Вихідні дані: *Allo* - сайт, що містить усі функції стандартного інтернет магазину. Під час використання сайту можна переглядати каталог товарів, дивитися їх характеристики та порівнювати між собою, увійти до акаунту, додавати товари до кошика, оформляти заявку, зв’язатися з підтримкою.

Мета: мета тестування сайту *Allo* є тест коректності роботи загального функціоналу сайту через прогін стандартних сценаріїв його використання.

У результаті тестування ми отримаємо інформацію про поточний стан працездатності сайту. Також буде зроблено звіт за результатами проведеного тестування.

Проведе двоетапне тестування, де спочатку буде зроблене ручне тестування а по його завершенню автоматизований варіант.

Умови тестування: Наш сайт має задовольняти усі потреби користувачів такі як: переглядати каталог товарів, дивитися їх характеристики та порівнювати між собою, увійти до акаунту, додавати товари до кошика, оформляти заявку, зв’язатися з підтримкою.

ОС на базі якої буде проведене тестування:

Windows 10

Браузер на базі якого буде проведене тестування:

Google Chrome

Типи тестування:

Функціональне тестування:

У результаті проведення даного тестування ми отримаємо звіт про наявність/відсутність функціональних помилок через виконання перевірки базованого на тестових сценаріях.

Це тестування буде проводитися за такою схемою:

Рисунок 3.1 - Схема тестування

Тестування кросбраузерності:

Перевірка сайту на візуальні помилки у дизайні. У нашому випадку перевірка буде відбуватися на базі лише 1 браузеру - *Google Chrome*.

Регресивне тестування:

Дана перевірка робиться під час розробки продукту на випадок що можуть з’явитися під час переходу сайту на нову версію.

Так як нами проводиться тестування готового продукту дана перевірка не обов’язкова до виконання, замість цього проведемо перевірку на працездатність.

Кінцевий результат:

Кінцевим результатом нашого тестування має стати звіт з проведених ручних та автотестів та рекомендації щодо можливого покращення функціоналу сайту.

3.4. Створення автотестів

Для проведення тестування нашого сайту створимо декілька тест-кейсів, що мають звичні для будь якого користувача сценарії використання сайту.

1. Відкриємо додаток *Katalon Studio* де створимо новий проєкт натиснувши кнопку “New Project”.

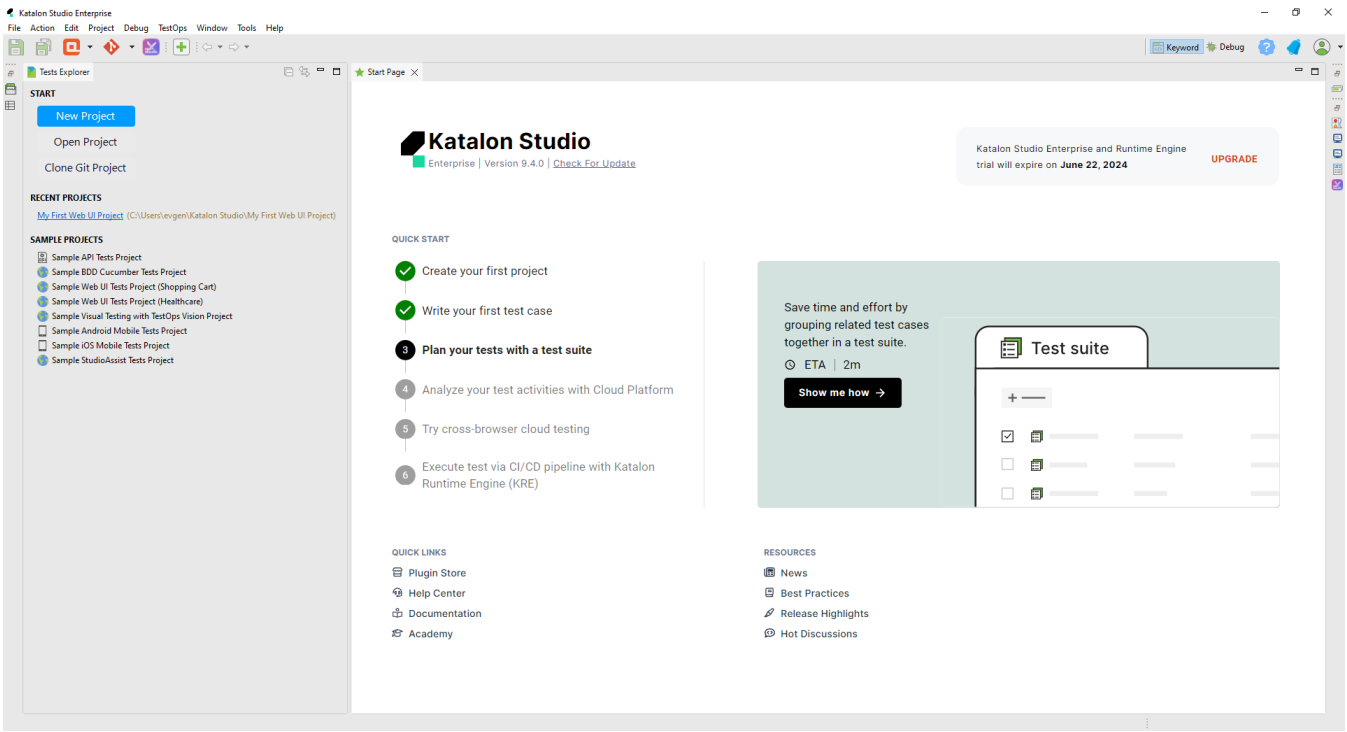


Рисунок 3.2 - Інтерфейс *Katalon Studio*

2. Зробимо налаштування нашого проєкту. Назвемо його “Test Allo”.

У вкладці розділі тип оберемо “Web” так як ми не будемо перевіряти на кросплатформленість.

Оберемо зручне місце розташування файлів нашого тесту.

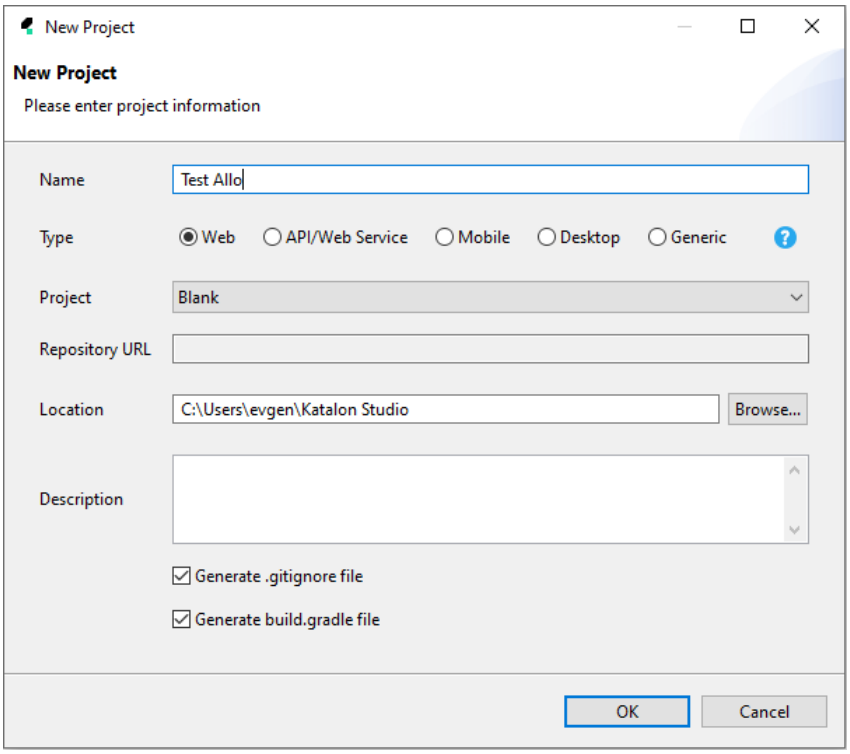


Рисунок 3.3 - Меню створення нового проєкту

3. Наступним кроком буде створення тест-кейсу зі входженням до аккаунту. Для цього ми скористаємося функцією запису дії користувача.

Дана функція дозволяє тестувальнику робити ручне тестування на базі якого *Katalon Studio* автоматично створює тест кейс зі скриптом. Дана функція значно спрощує процес автоматизації тестування.

Для цього потрібно натиснути “Record Web” після чого нам відкриється меню запису дії користувача де ми вносимо до строчки URL посилання на наш сайт. Далі нам потрібно натиснути клавішу “Record” після чого в обраному нами браузері відкриється наше посилання де ми будемо виконувати наші тест-кейси.

Наступним кроком буде створення першого тест-кейсу з позитивним сценарієм.

Тест-кейс №1

Метою даного тест-кейсу буде тестування системи входу до аккаунту користувача.

Попередньо користувач має бути зареєстрованим.

Таблиця. 3.1 Тест-кейс №1.

	Крок	Очікуваний результат
1	Відкрити сайт https://allo.ua/ua/	Відкривається головна сторінка
2	Натиснути кнопку ”Вхід”	Відкривається сторінка авторизації
3	Ввести у поле ”Номер телефона або e-mail” дійсний e-mail	Введений e-mail дійсний
4	Ввести у поле ”Пароль ” Дійсний пароль	Введений пароль співпадає
5	Натиснути кнопку ”Увійти”	Відкривається головна сторінка сайту

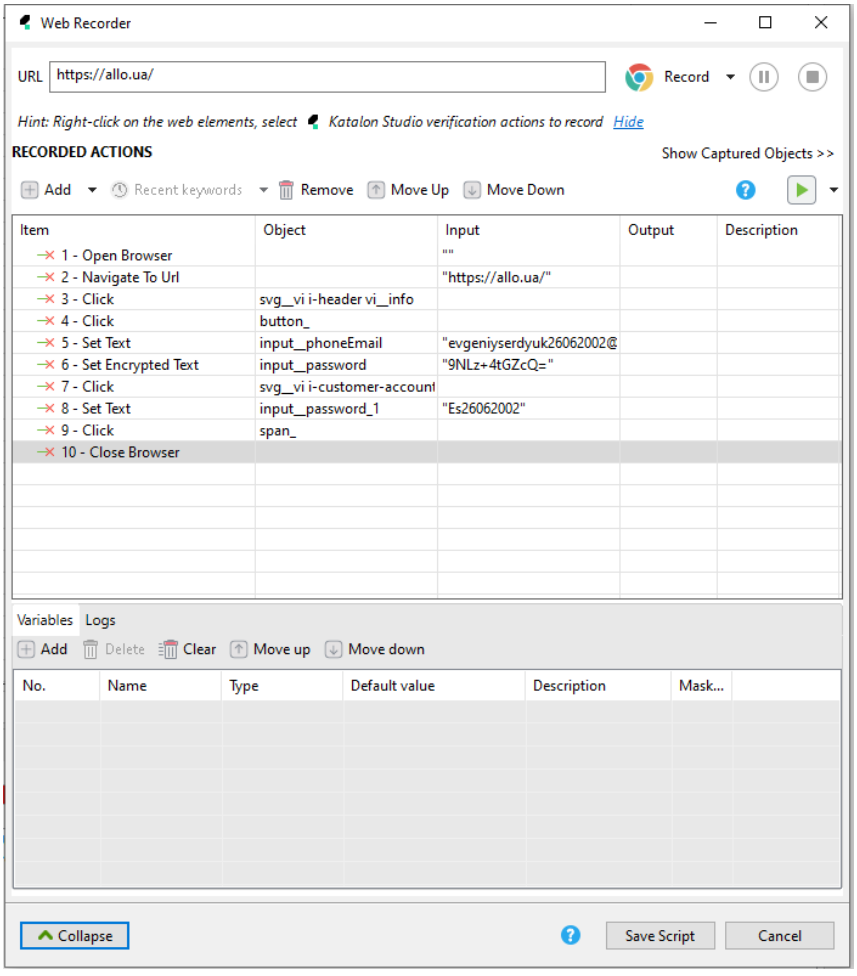


Рисунок 3.3 - Скрипт тест-кейсу №1

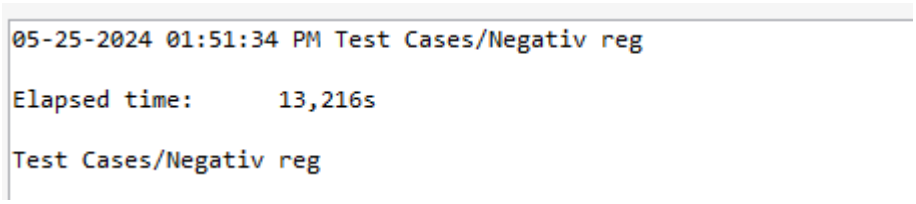


Рисунок 3.3 - Результат виконання тест-кейсу №1

Програма успішно провела автотест за позитивним сценарієм. Скрипт складався з 10 етапів. Усе тестування зайняло 13,216 сек.

Тест-кейс №2

Даний тест кейс буде повторювати скрипт першого ,за вийнятком, пункту з введенням правильного паролю. У цьому тест-кейсі ми виконаємо негативний сценарій, щоб отримати повідомлення про неправильний пароль чи електронну адресу.

Попередньо користувач має бути зареєстрованим.

Таблиця. 3.2 Тест-кейс №2.

	Крок	Очікуваний результат
1	Відкрити сайт https://allo.ua/ua/	Відкривається головна сторінка
2	Натиснути кнопку ”Вхід”	Відкривається сторінка авторизації
3	Ввести у поле ”Номер телефона або e-mail” дійсний e-mail	Введений e-mail дійсний
4	Ввести у поле ”Пароль ” Невірний пароль	Введений пароль співпадає
5	Натиснути кнопку ”Увійти”	Отримаємо повідомлення ”Неправильний e-mail або пароль ”

Item	Object	Input
1 - Open Browser		""
2 - Navigate To Url		"https://allo.ua/"
3 - Click	button__mh-button mh-button--ope	
4 - Click	span_	
5 - Set Text	input__phoneEmail	"evgeniyserdyuk26062002@gmail.co
6 - Set Encrypted Text	input__password	"9NLz+4tGZcQ="
7 - Click	svg__vi i-header vi__info	
8 - Double Click	input__password_1	
9 - Set Text	input__password_1	"123456"
10 - Click	span__1	

Рисунок 3.4 - Скрипт тест-кейсу №2

```
05-25-2024 02:46:50 PM Test Cases/Negativ reg

Elapsed time:      13,737s

Test Cases/Negativ reg
```

Рисунок 3.4 - Результат виконання тест-кейсу №2

У результаті проведення даного тесту з негативним сценарієм ми отримали вікно з оповіщення про неправильно введений пароль чи електронну адресу. Загальний час на виконання автотесту 13,737сек.

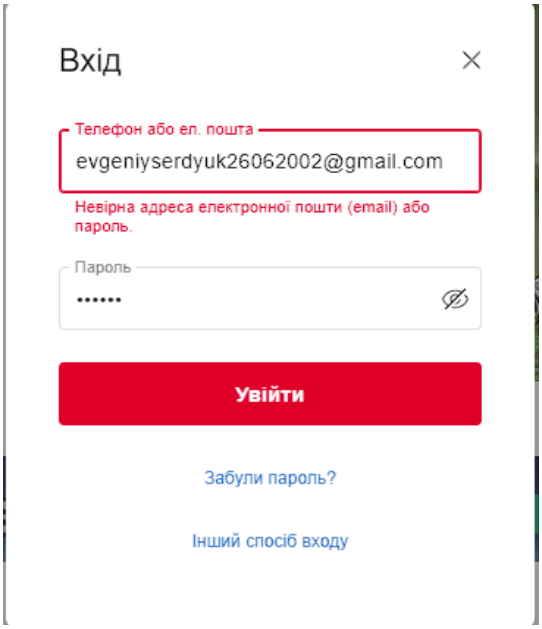


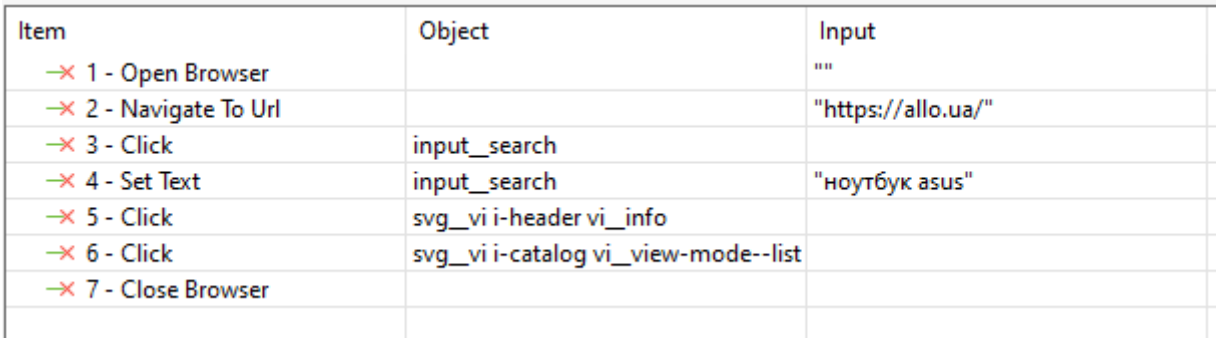
Рисунок 3.5 - Повідомлення про помилку

Тест-кейс №3

Основною функцією нашого сайту є можливість пошуку товару та його сортування за різними критеріями. Наступний тест-кейс буде орієнтований на тестування даних функцій сайту.

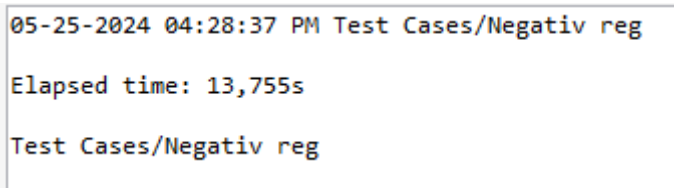
Таблиця. 3.3 Тест-кейс №3.

	Крок	Очікуваний результат
1	Відкрити сайт https://allo.ua/ua/	Відкривається головна сторінка
2	Натиснути кнопку ”пошук товарів”	У полі пошуку з’являється курсор
3	Ввести у поле пошуку ”Ноутбук Asus”	Наш текст з’являється у полі пошуку
4	Натиснути на кнопку пошуку	Відкривається сторінка з товарами що співпали нашим запитом
5	Натиснути кнопку ”Виводити” та оберемо варіант ”Від дешевих до дорогих”	Змінюється поряд показаних на екрані товарів



Item	Object	Input
1 - Open Browser		""
2 - Navigate To Url		"https://allo.ua/"
3 - Click	input__search	
4 - Set Text	input__search	"ноутбук asus"
5 - Click	svg__vi i-header vi__info	
6 - Click	svg__vi i-catalog vi__view-mode--list	
7 - Close Browser		

Рисунок 3.5 - Скрипт тест-кейсу №3



```
05-25-2024 04:28:37 PM Test Cases/Negativ reg

Elapsed time: 13,755s

Test Cases/Negativ reg
```

Рисунок 3.6 - Результат виконання тест-кейсу №3

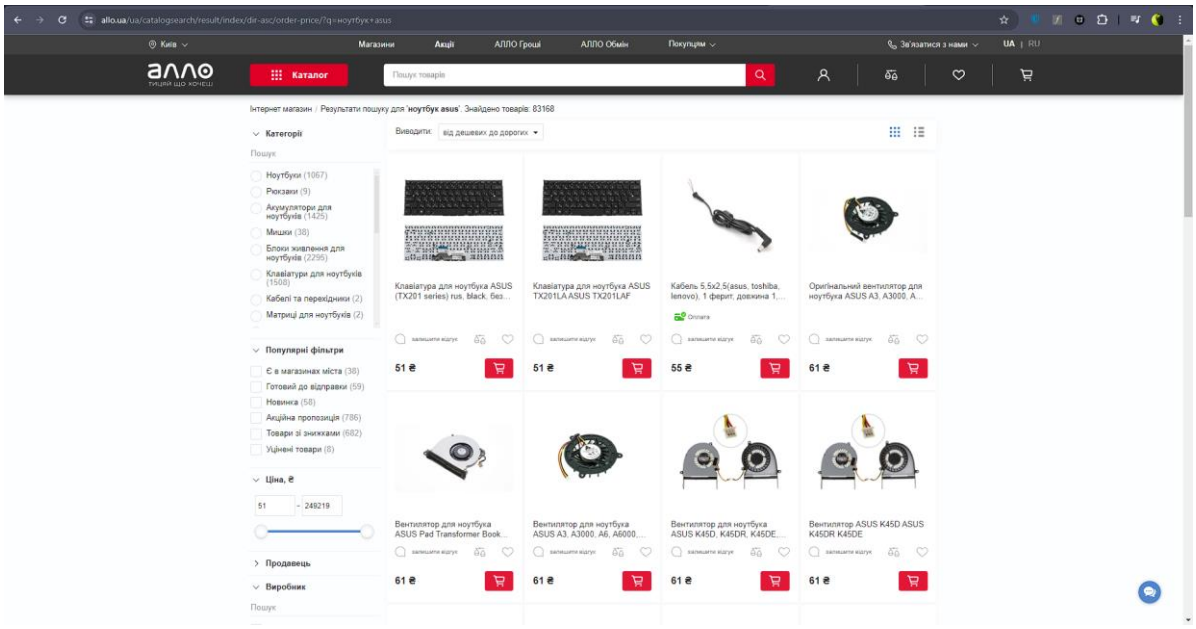


Рисунок 3.7 - Результат виконання тест-кейсу №3

Тест-кейс №4

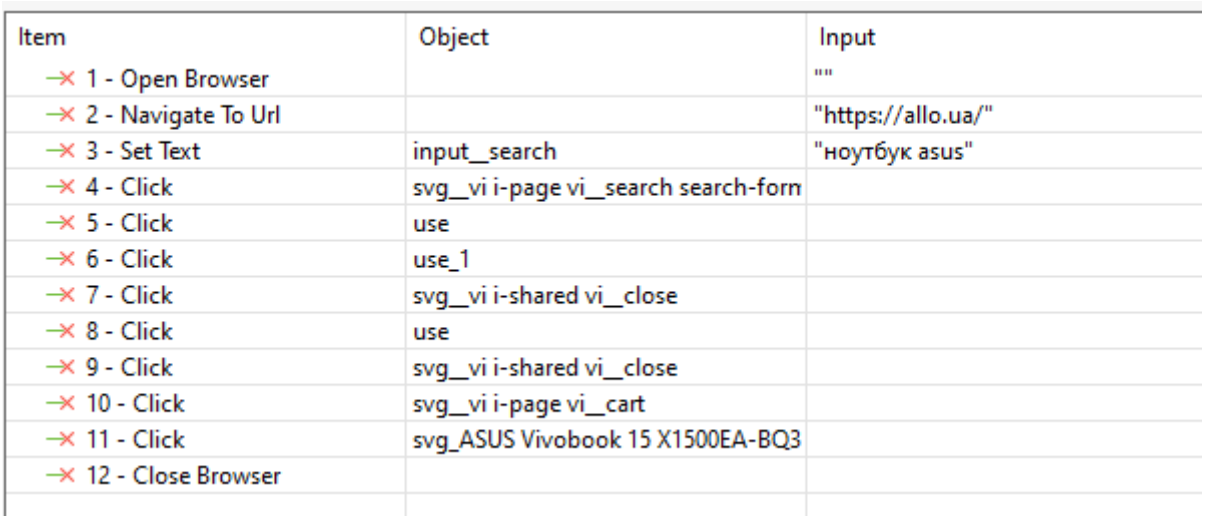
Також однією з найголовніших функцій інтернет магазинів є можливість додавати товар до кошику. Через пошук обираємо потрібний нам додаємо його та ще один до кошику, далі відкриваємо кошик та прибираємо другий з обраних товарів з нього.

Таблиця. 3.4 Тест-кейс №4.

	Крок	Очікуваний результат
1	Відкрити сайт https://allo.ua/ua/	Відкривається головна сторінка
2	Натиснути кнопку ”пошук товарів”	У полі пошуку з’являється курсор
3	Ввести у поле пошуку ”Ноутбук Asus”	Наш текст з’являється у полі пошуку
4	Натиснути на кнопку пошуку	Відкривається сторінка з товарами що співпали нашим запитом
5	Натиснути на перший та другий товари у списку	Відкриється сторінка з інформацією про товар

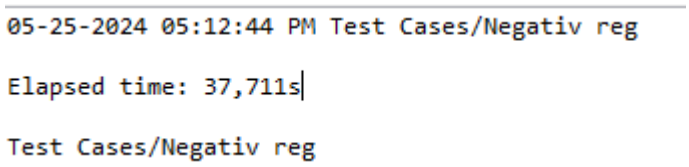
Продовження таблиці. 3.4 Тест-кейс №4.

6	Натиснути на ”Додати у кошик”	Товари додаються у кошик
7	Натиснути кнопку ”Кошик”	Відкриється кошик
8	Навести на другий обраний товар та натиснути ”Прибрати з кошика”	Товар прибирається з кошика



Item	Object	Input
1 - Open Browser		""
2 - Navigate To Url		"https://allo.ua/"
3 - Set Text	input__search	"ноутбук asus"
4 - Click	svg__vi i-page vi__search search-form	
5 - Click	use	
6 - Click	use_1	
7 - Click	svg__vi i-shared vi__close	
8 - Click	use	
9 - Click	svg__vi i-shared vi__close	
10 - Click	svg__vi i-page vi__cart	
11 - Click	svg_ASUS Vivobook 15 X1500EA-BQ3	
12 - Close Browser		

Рисунок 3.8 - Скрипт тест-кейсу №4



```
05-25-2024 05:12:44 PM Test Cases/Negativ reg

Elapsed time: 37,711s

Test Cases/Negativ reg
```

Рисунок 3.9 - Результат виконання тест-кейсу №4

3.5. Аналіз результатів тестування

Було проведено розроблено 4 тест-кейси серед яких є 3 з позитивним сценарієм та 1 з негативним. Тестування проводилося в 2 етапи. Спочатку ручне тестування потім на базі *Katalon Studio* будується автотест що повністю повторює виконаний ручний тест.

Автотест виконувався на ПК з наступними параметрами системи:

ОС – *Windows 10*

Процесор – *AMD Ryzen 5-3600 3,6G*

ОЗУ – 32Гб

Тип системи – 64-розрядна

Для більшого розуміння отриманих результатів проведемо порівняння часу витраченого на виконання роботи(таблиця 3.5, 3.6).

Таблиця. 3.5 Загальний час автотетсу.

№ тесту	Час виконання
1	13,216с.
2	13,737с.
3	13,755с.
4	37,711с.
сума	1хв. 18,419с.

Таблиця. 3.5 Загальний час ручного тесту.

№ тесту	Час виконання
1	27с.
2	26с.
3	35с.
4	54с.
сума	2хв. 12с.

Отриманий результат повністю підтвердив очікування. Було очевидно що автотест буде виконано набагато швидше. Для виконання обох типів тестування потрібна мінімальна кількість працівників. Використана нами програма є однією з найкращих на ринку сервісів автоматичного тестування, що дозволяє бути впевненим у результатах тестування.

На основі проведеного дослідження стало зрозуміло, що використання *Katalon Studio* для автоматизованого тестування є дуже ефективним з точки використаного на це часу. Також великим плюсом на користь даної програми можна вважати інтуїтивно зрозумілий інтерфейс, що дозволяє навіть не досвідченому користувачу розібратися в процесі автоматизації тестування.

Висновок

У даному розділі було розроблено систему автотестів до якої входять 3 тести з позитивним сценарієм та 1 з негативним. Було пояснено вибір інструмента автоматичного тестування. Було описано мету, ціль, етапи тестування. Були розроблені сценарії для тест кейсів, розписані передумови для їх виконання. Було виконано встановлення *Katalon Studio* та виконано його налаштування. Розроблені скрипти було реалізовано як у вигляді ручного тестування так і автоматичного. Проведено аналіз отриманих результатів та зроблені висновки проведеної роботи.

ВИСНОВКИ

Під час виконання дипломної роботи було розглянуте тестування як невід'ємна частина робочого процесу розробки ПЗ. Були проаналізовані типи, види та навпрямки тестування. Було розглянуте автоматичне тестування як один з основних напрямів спрощення роботи тестувальника. Детально проаналізували теорію тестування web-додатків, її критерії та особливості.

Були розглянуті найновітніші та найпопулярніші сервіси автоматизації тестування web-додатків. Проаналізували їх основні відмінності базуючись на критерії важливі для реалізації нашого проєкту. Розписали основні переваги та недоліки на основі яких обрали додаток що повністю задовольняє наші потреби.

Фінальним етапом написання дипломної роботи була розробка системи автотестів на основі отриманої раніше інформації. Було пояснено вибір інструмента автоматичного тестування. Описано мету, ціль, етапи тестування. Також були розроблені сценарії для тест кейсів, розписані передумови для їх виконання.

На основі проведеного дослідження було виявлено, шо автоматизація тестування дає преваги розробнику у вигляді можливості запуску тестів на будь яких пристроях у будь який час, що дозволяє одночасне проведення і ручного і автоматизованого тестування. Використання інструментів на кшталт *Katalon Studio* хоч і збільшує загальну вартість тестування проте також має свої перваги порівняно зі звичним нам ручним тестуванням. До таких переваг можна віднести значне скорочення часу тестування та полегшення самого процесу.

Програми по типу *Katalon Studio* великий функціонал, що покращує процес автоматизації тестування. Такі функції як використана нами можливість захвату екрану з подальшим записом дій користувача для

автоматичної побудови скриптів або візуальне тестування готової продукції на візульні баги та помилки з використанням штучного інтелекту допомагають роботи тестувальника ПЗ. Сучасні сервіси схожі на *Katalon Studio* роблять величезний вклад у роботи тестувальників допомагаючи полегшити та прискорити їх роботу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматизоване тестування. qalight. URL: https://qalight.ua/baza-znaniy/avtomatizovane-testuvannya/ (дата звернення: 12.02.2024).
2. Що таке автоматизація тестування? Без жаргонів, простий посібник. Zaptest. . URL: https://www.zaptest.com/uk/що-таке-автоматизація-тестування-без (дата звернення: 16.02.2024).
3. Куликов Святослав Тестирование программного обеспечения. Базовый курс/ Святослав Куликов., 2021. – Т. 3 : – 298 с.
4. Burns D. Selenium 2 Testing Tools: Beginner’s Guide / Burns D. – Birmingham: Packt Publishing, 2012. – 437 с.
5. Криспин Л., Грегори Д. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams / Л. Криспин, Д. Грегори. – М. : Вильямс, 2010. – 464 с.
6. Канер К., Фолк Д., Нгуен Е. К. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / К. Канер, Д. Фолк, Е. К. Нгуен. — Киев : ДиаСофт, 2001. — 544 с.
7. Що таке тестування пз, його етапи, види,інструменти. Sigma Softaware University. URL: https://university.sigma.software/what-is-software-testing/ (дата звернення: 05.03.2024).
8. An all-in-one test automation solution URL: https://www.katalon.com/ (дата звернення: 07.03.2024).
9. Webdriver. Selenium.URL: https://www.selenium.dev/documentation/webdriver/ (date of access: 16.03.2024).

10. Move faster and reduce costs with the power of AI. Testim. URL: *https://www.testim.io/* (date of access: 21.03.2024).

11. Test.Automate.Accelerate. Cypress.io. URL: *https://www.cypress.io/* (дата звернення: 04.04.2024).

12. Як створити веб-додаток: типи, переваги, принцип роботи. Wezom. . URL: *https://wezom.com.ua/ua/blog/kak-sozdat-veb-prilozhenie* (дата звернення: 10.04.2024).

13. Що таке веб додаток? різниця між сайтом, веб-додатком, spa і pwa. Webcase. URL: *https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/* (дата звернення: 18.04.2024).

14. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects URL *http://tisten.ir/wp-content/uploads/2019/01/Complete-Guide-to-Test-Automation-Techniques-Practices-and-Patterns-for-Building-and-Maintaining-Effective-Software-Projects-Apress-2018-Arnon-Axelrod.pdf*

15. У чому відмінність сайту від веб-додатка. Foxminded. URL*: https://foxminded.ua/chym-vidrizniaietsia-sait-vid-veb-dodatku/* (дата звернення: 26.04.2024).

16. Raj B. S., Panigrahi S. Selenium WebDriver Practical Guide / B. S. Raj, S. Panigrahi. – Packt Publishing, 2013.

17. Goucher A., Riley T. Beautiful Testing: Leading Professionals Reveal How They Improve Software / A. Goucher, T. Riley. – O'Reilly Media, 2009.

18. Bach J., Bolton M. Rapid Software Testing / J. Bach, M. Bolton. – Context-Driven Testing, 2013.

19. Jones E. Effective Software Testing: 50 Specific Ways to Improve Your Testing / E. Jones. – Addison-Wesley Professional, 2009.

20. Fewster M., Graham D. Software Test Automation / M. Fewster, D. Graham. – Addison-Wesley, 1999.

21. Nguyen T., Kaner C., Falk J. Testing Computer Software / T. Nguyen, C. Kaner, J. Falk. – Wiley, 1999. – 608 с.

22. Marick B. The Craft of Software Testing: Subsystem Testing Including Object-Based and Object-Oriented Testing / B. Marick. – Prentice Hall, 1995. – 368 с.