

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Кафедра** \_\_\_\_\_ **Комп'ютерних систем та мереж** \_\_\_\_\_

**ДОПУСТИТИ**  
**ДО ЗАХИСТУ**  
Завідувач кафедри  
комп'ютерних систем та  
мереж

\_\_\_\_\_ (Жуков І.А.)

« \_\_\_\_ » \_\_\_\_\_ 2021 р.

**ДИПЛОМНИЙ ПРОЄКТ**  
**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ**  
**«БАКАЛАВР»**

**Тема:** \_\_\_\_\_ Система автоматизованого тестування *Web*-додатків \_\_\_\_\_

**Виконавець:** \_\_\_\_\_ Белозьорова Д.О. \_\_\_\_\_

**Керівник:** \_\_\_\_\_ Журавель С.В. \_\_\_\_\_

**Нормоконтролер:** \_\_\_\_\_ Журавель С.В. \_\_\_\_\_

**Київ 2021**

# НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Напрямок (спеціальність) 123 «Комп'ютерна інженерія»

(шифр, найменування)

ЗАТВЕРДЖУЮ

Завідувач кафедри  
комп'ютерних систем та мереж

\_\_\_\_\_ (Жуков І.А.)

«\_\_\_» \_\_\_\_\_ 2021 р.

## ЗАВДАННЯ на виконання дипломного проєкту

Белозьоровій Діані Олегівній

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проєкту (роботи): Система автоматизованого тестування *Web*-додатків

затверджена наказом ректора від «26» квітня 2021 року № 648/ст.

2. Термін виконання проєкту (роботи): з 24.05.2021 до 20.06.2021

3. Вихідні дані до проєкту (роботи): інформація про тестування, автоматизоване тестування, інструменти для процесу розробки.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) Теорія тестування та види.

2) Автоматизоване тестування переваги та недоліки.

3) Інструменти для автоматизації. (*Selenium*).

4) Система автотестів на основі тест-кейсів.

5. Перелік обов'язкового графічного матеріалу:

Презентація *PowerPoint*

## 6. Календарний план

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів	Примітка
1	Ознайомитись з темою та постановкою задачі дипломного проєкту	24.05.2021	Виконано
2	Вивчити спеціальну літературу і технічну документацію	25.05.2021 – 26.05.2021	Виконано
3	Проаналізувати основи автоматизованого тестування <i>Web</i> -додатків	27.05.2021 – 28.05.2021	Виконано
4	Написати перший розділ щодо аналізу предметної області	29.05.2021 – 30.05.2021	Виконано
5	Проаналізувати інструменти, переваги та їх недоліки	31.05.2021	Виконано
6	Написати другий розділ щодо аналізу інструментів та їх встановлення	01.06.2021 – 03.06.2021	Виконано
7	Створити тест план та тест-кейси і описати автотести	04.06.2021 – 06.06.2021	Виконано
8	Написати третій розділ про процес підключення програм та проєктування тестів	07.06.2021 – 09.06.2021	Виконано
9	Оформлення пояснювальної записки	10.06.2021 – 12.06.2021	Виконано
10	Підготувати графічний та демонстраційний матеріал	13.06.2021	Виконано

7. Дата отримання завдання «24» травня 2021 р. \_\_\_\_\_

Керівник дипломного проєкту \_\_\_\_\_ Журавель С.В.  
(підпис)

Завдання прийняв до виконання \_\_\_\_\_ Белозьорова Д.О.  
(підпис студента)

## РЕФЕРАТ

Пояснювальна записка до дипломного проєкту на тему «Система автоматизованого тестування *Web*-додатків»: 57 с., 17 рис., 6 таблиць, 20 літературних джерел, 1 додаток.

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ, АВТОТЕСТИ, *SELENIUM*, *JAVA*.

**Мета дипломного проєкту** – проаналізувати теорію автоматизованого тестування веб-додатків та створити систему за допомогою автотестів.

**Завдання дипломного проєкту** – створити систему автоматизованого тестування *web*-додатка. Проаналізувати документацію, розробити тест-кейси. Описати скрипти на мові програмування *Java* в інструменті *Selenium*.

**Об'єкт проєктування** – система автотестів *Web*-додатків.

**Програмне забезпечення для виконання завдання:** *Selenium Webdriver*, *Intellij IDEA*.

Автоматизоване тестування, Документація, Тест-план, Тест-кейс, Скрипт, *Java*, *Selenium*, *Intellij IDEA*.

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

## ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ .....	7
ВСТУП .....	8
РОЗДІЛ 1 АСПЕКТИ СУЧАСНИХ ПІДХОДІВ ТА МЕТОДІВ	
ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....	10
1.1. Сутність поняття тестування програмного забезпечення .....	10
1.2. Типи тестування .....	11
1.3. Види і направлення тестування .....	13
1.4. Автоматизоване тестування .....	15
1.5. <i>Web</i> -додатки .....	18
1.6. Архітектура клієнт-серверного додатку .....	18
Висновки до розділу .....	21
РОЗДІЛ 2 ІНСТРУМЕНТИ ДЛЯ АВТОМАТИЗОВАНОГО	
ТЕСТУВАННЯ <i>WEB</i> -ДОДАТКІВ .....	22
2.1. Критерії вибору інструменту .....	22
2.2. Топ 3 інструменти в 2021 році .....	23
2.3. Характеристика інструменту <i>Selenium</i> .....	25
2.4. Характеристика інструменту <i>Katalon Studio</i> .....	28
2.5. Характеристика інструменту <i>UFT</i> .....	29
Висновки до розділу .....	32

## РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО

ЗАБЕЗПЕЧЕННЯ .....	33
3.1. Опис <i>Web</i> -додатка .....	33
3.2. Необхідні інструменти для створення автотестів .....	34
3.3. Вибір мови програмування .....	35
3.4. Відображення процесу проходження документації .....	37
3.5. Автоматизоване тестування системи <i>Web</i> -додатка .....	42
3.6. Аналіз отриманих результатів .....	49
Висновки до розділу .....	53
ВИСНОВКИ .....	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	55
ДОДАТКИ	

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

**ПК** – персональний комп'ютер.

**ПЗ** – програмне забезпечення.

**IDE** – середовище розробки.

**ОС** – операційна система.

**HTTP** – *Hypertext Transfer Protocol*.

**HTTPS** – *HyperText Transfer Protocol Secure*.

**API** – прикладний програмний інтерфейс.

**GUI** – графічний інтерфейс користувача.

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

**Selenium** – набір інструментів, призначених для автоматизації веб браузерів на різних платформах.

## ВСТУП

Ще за старих часів тестування програмного забезпечення мало на меті пошук помилок у продукті. Мета полягає в тому, щоб поліпшити якість продукції. Але в наш час діапазон тестування програмного забезпечення розширився.

У 2021 році попит на експертів з контролю якості, що володіють конкретними знаннями в області тестування, зростає, і це буде однією з основних тенденцій тестування автоматизації. Тестування автоматизації призвело до величезних зрушень у тестуванні програмного забезпечення. Професіонали з контролю якості змогли успішно засвідчити підвищену ефективність тесту та збільшення охоплення тестом, а також нові можливості у виконанні тестів, яких ніколи не було при ручному тестуванні.

Автоматизація тестування розвивається в результаті розвитку штучного інтелекту. Тестування автоматизації спрямоване на зменшення ручних зусиль для повторюваних процесів, таких як регресійне тестування. Але навіть написання тестових кейсів та сценаріїв вимагає ручних зусиль. Крім того, автоматизація тестування не замінить повністю ручну взаємодію.

Отже, тенденція до 2021 року – це не автоматизація тестування, а пошук правильного балансу між тестуванням вручну та автоматизацією.

Наукова значимість: полегшене та прискорене тестування web-додатків за допомогою автоматизації.

Мета дипломної роботи – проаналізувати теорію автоматизованого тестування веб-додатків та створити автотести на основі тест-кейсів за допомогою інструменту *Selenium WebDriver*.

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



Розглянемо інструментарії та технології, які можуть бути використані для автоматизації тестування.

Для досягнення даної мети в дипломному проєкті 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 автотестів.
7. Провести аналіз результатів.

# **РОЗДІЛ 1**

## **АСПЕКТИ СУЧАСНИХ ПІДХОДІВ ТА МЕТОДІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

### **1.1. Сутність поняття тестування програмного забезпечення**

Тестування – це одна з технік контролю якості, що включає в себе планування, проєктування, виконання тестування та аналіз відповідних результатів.

Тестування програмного забезпечення – це процес аналізу програмного засобу і відповідної документації, з метою виявлення дефектів і підвищення якості продукту.[4]

Тестування ПЗ охоплює не тільки проведення тестів, але й інші елементи процесу забезпечення якості, такі як:

1. Планування та аналіз вимог.
2. Критерії початку тестування.
3. Стратегія тестування.
4. Проєктування тестових сценаріїв (тест-планів, тест-кейсів, користувацьких вимог).
5. Виконання тест-кейсів, вимог.
6. Фіксація дефектів.
7. Аналіз результатів.
8. Написання звітів.

Стосовно цілі тестування, можна виділити такі основні аспекти:

- Надання інформації про якість програмного забезпечення кінцевому замовнику.

- Підвищення якості тестового ПЗ.
- Запобігання виявленню нових дефектів.

## 1.2. Типи тестування

**Ручне тестування (*manual testing*)** – це вид тестування, в якому тестують ПО вручну, тобто не використовують ніяких середовищ автоматизації.

Тестувальник має роль звичайного користувача програми та перевіряє продукт, щоб виявити баги у програмі. Ручне тестування – найбільш низькорівневий і простий тип тестування, який може займати багато часу, але якщо глянути на довгострокову перспективу, то він економічно вигідніший. [4]

Ручне тестування поділяється на декілька видів, як-от:

1. Модульне тестування ( *Unit Testing*).
2. Інтеграційне тестування ( *Integration Testing*).
3. Системне тестування ( *System Testing*).
4. Операційне тестування ( *Release Testing*).
5. Приймальне тестування ( *Acceptance Testing*).

Для професійного тестування використовується тест-плани з варіантами тестування.

Тест-план – це документ, який описує весь розмір роботи, яку повинен виконати тестувальник, починаючи з опису об'єкту, цілі, ресурси і графіки запланованих тестових активностей, стратегії тестування, розкладу, критерії початку і кінця тестування, до необхідного в процесі тестування обладнання, методи проєктування тестів. Оцінка всіх можливих ризиків тестування з варіантами їх можливого вирішення. [4]

Переваги такого виду тестування полягає в таких аспектах:

- Користувацький *feedback*. Весь звіт тестувальника можна розглянути як відгук від повноцінного користувача.
- Інтерфейсний *feedback*. Користувацький інтерфейс можна протестувати повністю тільки в ручну.
- Економічна вигода.

- Тестування в реальному часі. Можливість розпочати тестування на перших етапах впровадження без кінцевого продукту.
- Можливість дослідницького тестування.

Недоліки такого тестування:

- Людський фактор. Це найбільш впливовий фактор, тому що усі люди можуть допустити помилки.
- Неможливість перевірки навантажувального тестування.
- Важкість повторної перевірки після внесення змін.

**Автоматизоване тестування (*automated Testing*)** – це набір технік, підходів і інструментальних елементів, які дозволяють вилучити тестувальника з виконання деяких завдань в процесі тестування.[4]

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).[4]

Для професійного тестування використовуються тест-кейси, які частково або повністю покривають інструментальне середовище, однак розробка тест-кейсів, запуск, оцінка виконання тестів та опис дефектів в баг-трекінгових системах не обходиться без втручання тестувальника.

Тест-кейс – набір вхідних даних, умови виконання, очікуваний і фактичний результат, розроблений з цілю перевірки властивостей і поведінку програмного середовища.

Переваги такого виду тестування:

- Тестування навантаженості системи.
- Час виконання автотестів.
- Легкість повторної перевірки після внесення змін.
- Повторюваність.

Недоліки такого тестування:

- Великі витрати.

- Вартість інструменту автоматизації.
- Пропуск дрібних помилок.

**Напівавтоматизоване тестування (*semiautomated Testing*)** – використовуючи цей вид тестування, частину проводимо в ручному тестуванні, а іншу половину ми автоматизуємо за допомогою тестів.

### 1.3. Види і напрямлення тестування

Тестування можна класифікувати за дуже великою кількістю ознак. Розглянемо список спрощеної класифікації тестування (рис. 1.1).[4]

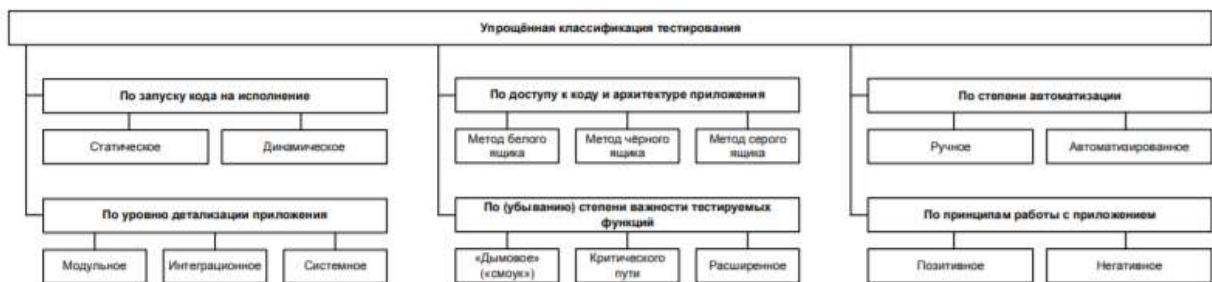


Рис. 1.1. Спрощена класифікація тестування.

Перший вид – це тестування по запуску коду на виконання їх є два це:

- Статичне тестування – без запуску коду.
- Динамічне тестування – з запуском коду.

До статичного тестування відносяться:

1. Документи (вимоги користувачькі, бізнес вимоги, тест-кейси, користувачькі історії, база даних).
2. Графічні прототипи.
3. Код програми.
4. Параметри програми.
5. Підготовка тестових даних.

Динамічне тестування передбачає роботу з кодом, подільним кодом, його частинами, перевірку поведінки програми.

Далі по доступу до коду і самої програми:

- Метод білого ящика – доступ до коду є.
- Метод сірого ящика – тільки до частини коду є доступ.
- Метод чорного ящика – без доступу до коду.

За ступенем автоматизації:

- Ручне тестування.
- Автоматизоване тестування.

За рівнем деталізації додатку:

- Модульне тестування.

Перевіряє функціональність і шукає дефекти в частинах додатка, які доступні і можуть бути протестовані окремо.

- Інтеграційне тестування.

Перевіряється взаємодія між компонентами системи після проведення компонентного тестування.

- Системне тестування.

Перевірка як функціональних, так і нефункціональних вимог в системі в цілому.

За ступенем важливості тестових функцій:

- Димове тестування;

Перевіряються ключові функціональності, без яких сам додаток не має важливості.

- Тестування критичного шляху;

Перевіряються найбільш важливі елементи і функції програми, правильність роботи та їх використання.

- Розширене тестування;

Перевірка усієї іншої функціональності.

За принципом роботи з додатком:

- Позитивне тестування;

Це тестування, яке перевіряє поведінку нашої програми на звичайне, валідне користування. Введення даних, які відповідають дійсності.

- Негативне тестування.

Це тестування, яке перевіряє поведінку програми на не звичні для нього команди. Введення даних, які не відповідають дійсності. Перевірка виводу помилок, і перевірка того, як поведе себе даний додаток.

Класифікація за природою додатка:

- Тестування *web*-дodatка.

Тестування пов'язане з інтенсивною діяльністю в області тестування сумісності, тестування продуктивності.

- Тестування мобільних додатків;

Також вимагає підвищеної уваги до тестування сумісності, оптимізації продуктивності, автоматизації тестування із застосуванням емуляторів мобільних пристроїв.

- Тестування десктопних додатків. [4]

Є найбільш класичним з цих 3 класифікацій, і його особливості залежать від предметної області додатків, нюансів архітектури, ключових показників якості і т.д.

#### **1.4. Автоматизоване тестування**

Автоматизація тестів – найкращий спосіб підвищити ефективність, охоплення тестом і швидкість виконання при тестуванні програмного забезпечення. [1]

Автоматизоване тестування програмного забезпечення важливо з таких причин:

- Ручне тестування всіх робочих процесів, усіх полів, усіх негативних сценаріїв вимагає часу та грошей.
- Складно перевірити багатомовні сайти вручну.
  - Автоматизація тестів при тестуванні програмного забезпечення не вимагає втручання людини. Ви можете запустити тест і залишити його без нагляду.

- Автоматизація тестів збільшує швидкість виконання тесту.
- Автоматизація сприяє збільшенню охоплення тестом.
- Тестування вручну може стати нудним, схильним до помилок.

Автоматизований процес тестування:

У процесі автоматизації виконуються наступні кроки (рис. 1.2):

- 1) Вибір інструменту тестування.
- 2) Визначення сфери автоматизації.
- 3) Планування, проєктування та розробка.
- 4) Виконання тесту.
- 5) Технічне обслуговування.

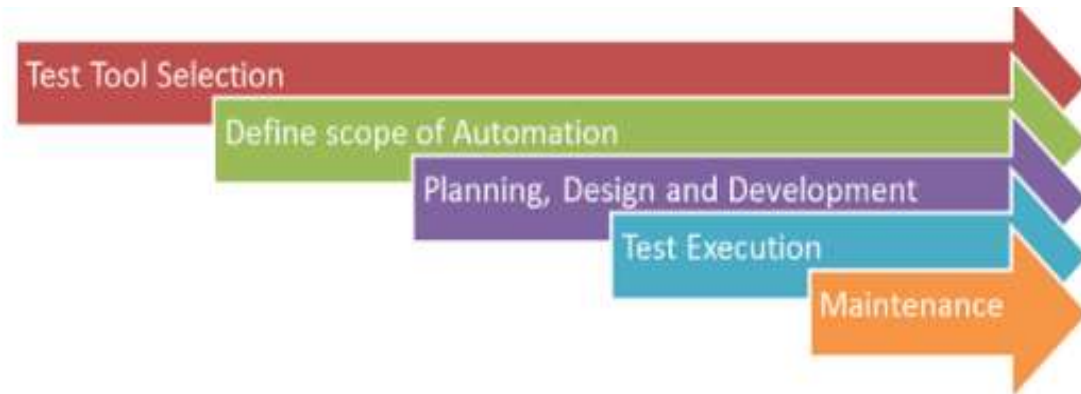


Рис. 1.2. Процеси автоматизованого тестування.

Види автоматизованого тестування:

- Димове тестування.

Тестування, що проводиться на початковому етапі і в першу чергу спрямоване на перевірку готовності розробленого продукту до проведення більш розширеного тестування, визначення загального стану якості продукту.

- Модульне тестування.

Це вид тестування програмного забезпечення, де тестуються окремі блоки або компоненти програмного забезпечення. Метою є перевірити, що кожна одиниця програмного коду працює належним чином. Модульне тестування проводиться під час розробки програми розробниками.

- Інтеграційне тестування.



Визначається як тип тестування, де програмні модулі інтегровані логічно та перевіряються як група. Типовий програмний проєкт складається з безлічі програмних модулів, кодованих різними програмістами. Метою цього рівня тестування є виявлення дефектів взаємодії між цими програмними модулями при їх інтеграції.

- Функціональне тестування.

Вид тестування, спрямований на перевірку коректності роботи функціональності додатку (коректність реалізації функціональних вимог). Часто функціональне тестування асоціюють з тестуванням методом чорного ящика, однак і методом білого ящика цілком можна перевіряти коректність реалізації функціональності.

- Тестування на основі ключових слів.

Спеціалізований підхід, згідно з яким використовуються деякі ключові слова, детально описують набір виконуваних дій, які, так чи інакше, потрібні для проходження певного етапу тестового сценарію.

- Регресійне тестування.

Визначається як тип тестування програмного забезпечення для підтвердження того, що нещодавня зміна програми або коду не вплинула негативно на існуючі функції.

- Перевірка рівня даних.

Сконцентровано на тій частині програми, яка відповідає за зберігання і деяку обробку даних (найчастіше – в базі даних чи іншому сховищі). Тут особливий інтерес представляє тестування даних, перевірка дотримання бізнес-правил, тестування продуктивності.

- Тестування чорної скриньки.

Це техніка, що використовується для перевірки функціональності програмного забезпечення, що працює виключно із зовнішнім інтерфейсом. Цей метод тестування також називають поведінковим тестуванням та функціональним тестуванням.

## **1.5. Web-додатки**

*Web*-додаток – клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера.

Програмне забезпечення веб-додатків складається з програм та даних, призначених для роботи в усьому світі, поєднання мережевого та клієнт-серверного програмного та апаратного забезпечення, виконуючи операції між локальними та віддаленими користувачами комп'ютера.

Для того, щоб найкраще визначити, як тестувати веб-програмне забезпечення, як і будь-яке інше програмне забезпечення, тестери повинні розуміти характеристики веб-програмного забезпечення, поведінку загалом та її взаємодію з іншим програмним та апаратним забезпеченням.

*Web*-програмне забезпечення, як і будь-яке інше програмне забезпечення, має особливий потенціал, проблемні місця, що призведе до збою програми або системи, якщо програма вийде з ладу, нам необхідно передбачити такі місця та впоратись із проблемами.

## **1.6. Архітектура клієнт-серверного додатку**

Клієнт-серверна архітектура – це архітектура, в якій мережеве навантаження розділяється між постачальниками послуг (серверами) і замовниками (клієнтами).[6]

Клієнт і сервер це програмне забезпечення розміщене на різних розрахункових машинах, що взаємодіє через мережеві протоколи. Можуть бути розміщені на одній машині (рис. 1.3).

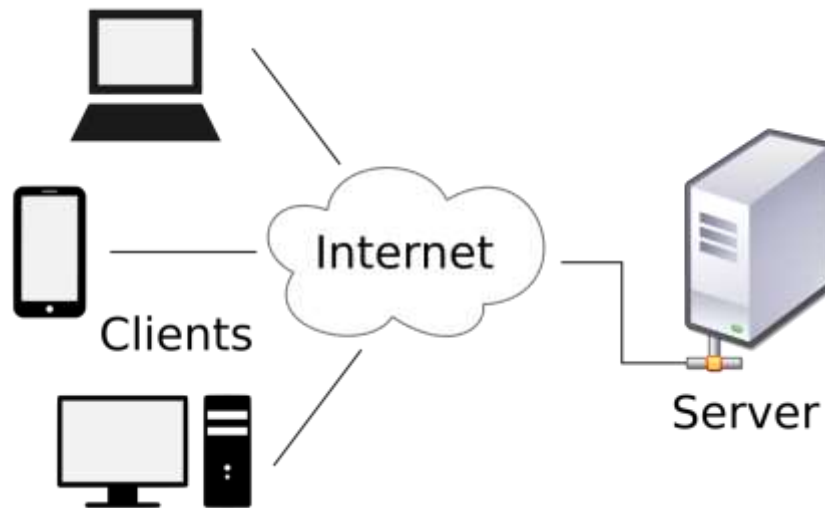


Рис. 1.3. Клієнт-серверна архітектура.

У тестуванні розрізняють типи Клієнт-браузера:

- Тонкий – пк або програма, яка переносить обробку інформації на сервер (Браузер).
- Товстий – додаток, який забезпечує розширену функціональність в незалежності від сервера (Сховище даних, 1С бухгалтерія, усі онлайн-ігри).

Переваги та недоліки клієнт-серверної архітектури:

- + Відсутність дублювання коду програми сервера і клієнта.
- + Всі дані на сервері і він більш захищеніший.
- + Простіше контролювати доступ.
- Вартість
- Залежність від роботи сервера.

Запит посилає клієнт на сервер, а сервер реагує на запит (*request*) і віддає свій *Http response* (відповідь).

Запит складається з таких компонентів як:

- Метод;

Методи поділяються на (*GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATH*)

- Версія протоколу;

Поділяються на *HTTP/1.1, HTTP/2*(Більш захищеніший)

- Хост машини;

Місце знаходження ресурсу. (*URL*)

- Хедер.

Додаткова інформація, яку містить відправлений ресурс (рис. 1.4).[7]

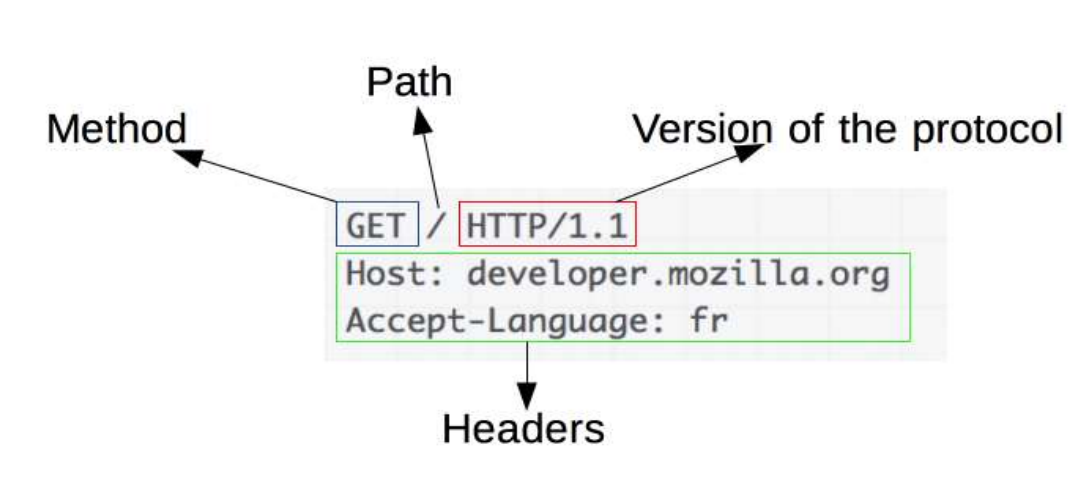


Рис. 1.4. *Http request.*

Відповідь на запит складається з таких компонентів як:

- Версія протоколу;

Поділяються на *HTTP/1.1*, *HTTP/2* (більш захищеніший)

- Статус стану коду;

Вказує на успішність або неуспішність запиту.

Поділяється на:

1xx – переадресація, 100 – вказує, що браузер відправив запит на сервер, але процес відправки ще продовжується.

2xx – такі коди інформують про те, що сервер прийняв запит браузера і процес пройшов успішно.

3xx – з'являється, коли сервер вказує, що сторінка, яку ми запрошуємо, тимчасово або назавжди видалена та ресурс не доступний.

4xx – такі повідомлення свідчать про помилку у запиті.

5xx – такі повідомлення свідчать про помилку на сервері.

- Статус повідомлення;

Опис стану коду.

- Хедер.

Додаткова інформація, яку містить відправлений ресурс (рис. 1.5).[7]

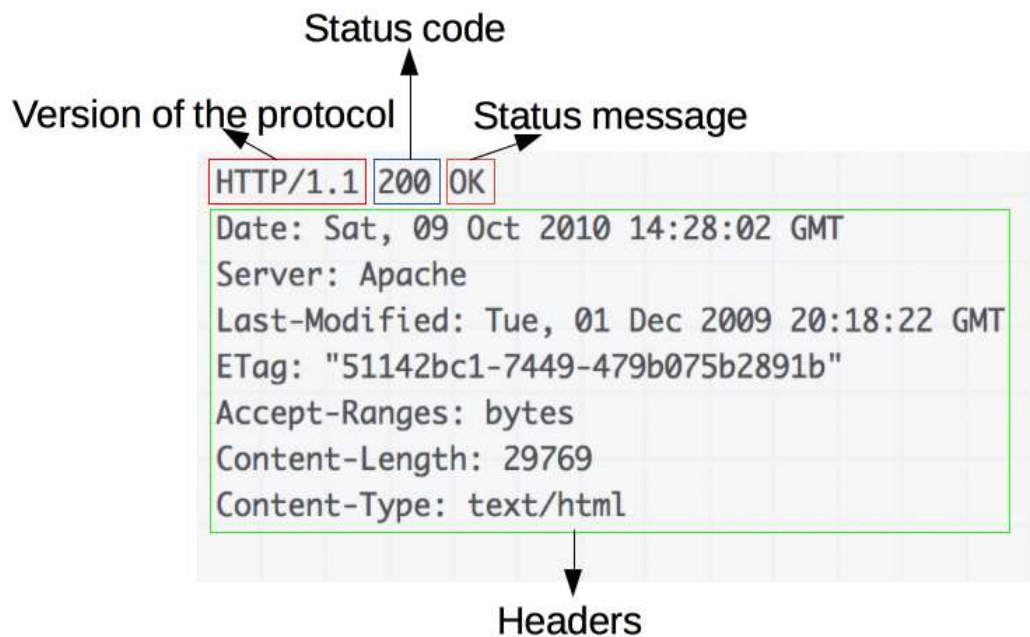


Рис. 1.5. *Http response.*

## Висновки до розділу

У першому розділі було розкрито теоретичну основу тестування ПЗ. Розкрили сутність поняття тестування, типи тестування, які можна використати, їх види, які детально описали та структурували. Описали поняття Web-додатку, його сутність та особливості. Проаналізовано переваги та недоліки різних підходів до тестування. Також охарактеризували архітектуру клієнт-серверного додатку, види запитів та їх детальний розбір. Наступним кроком буде опис інструментів для автоматизованого тестування.

## РОЗДІЛ 2

### ІНСТРУМЕНТИ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ WEB-ДОДАТКІВ

#### 2.1. Критерії вибору інструментів

Вибір правильного інструменту може бути складним завданням. Наступний критерій допоможе нам обрати найкращий інструмент для нашого автоматизованого тестування:

- Підтримка навколишнього середовища.
- Простота використання.
- Тестування бази даних.
- Ідентифікація об'єкта.
- Тестування зображень.
- Тестування відновлення помилок.
- Картографування об'єктів.
- Використовувана мова сценаріїв.
- Підтримка різних типів тестів – включаючи функціональні, управління тестами, мобільні тощо.
- Підтримка декількох фреймворків тестування.
- Легко налагоджувати сценарії програмного забезпечення для автоматизації.
- Здатність розпізнавати предмети в будь-якому середовищі.
- Великі звіти та результати випробувань.
- Мінімізована вартість навчання вибраних інструментів.

Вибір інструменту – одна з найбільших проблем, яку потрібно вирішити, перш ніж йти на автоматизацію.

По-перше, визначаємо вимоги, вивчаємо різні інструменти та їх можливості, встановлюємо очікування від цього інструменту та переходимо до підтвердження концепції.

На ринку доступно безліч інструментів для функціонального тестування. Розглянемо найбільш популярні з них.

## 2.2. Топ 3 інструмента у 2021 році

Засоби автоматизації тестування – це програмні інструменти, які допомагають користувачам тестувати різні настільні, веб- та мобільні програми. Ці інструменти надають рішення для автоматизації з метою автоматизації процесу тестування. Автоматизовані засоби тестування також пропонують безліч функцій для тестування графічного інтерфейсу, тестування продуктивності, тестування навантаження та тестування *API*. [1]

Нижче наведено кілька найкращих інструментів автоматизації випробувань:

1. *Selenium*.
2. *Katalon Studio*.
3. *UFT One*.

Ось найкращі засоби автоматизації тестів, які, як вважають, найкраще вирішують проблеми автоматизації протягом кількох років. Інструменти, включені до цього списку, обираються за цими критеріями:

- Підтримка тестування *API* та сервісів.
- Пропонуючи деякі можливості *AI / ML* та аналітики.
- Популярність і завершеність.

Розглянемо кожен з них, детальніше оцінимо їхні параметри, переваги та недоліки роботи з інструментом (таблиця 2.1).

Таблиця 2.1

Таблиця критеріїв інструментів

Продукт	<i>Selenium</i>	<i>Katalon Studio</i>	<i>UFT One</i>
Рік випуску	2004	2015	1998
Види тестування	<i>Web app</i>	<i>Web/API/Mobile/Desktop apps</i>	<i>Web/Desktop/Mobile/RPA apps</i>
Підтримувальні системи	<i>Windows Mac OS Linux Solaris</i>	<i>Windows Linux OS X</i>	<i>Windows</i>
Підтримувальні мови	<i>Java, C#, Perl, Python, JS, Ruby, PHP</i>	<i>Java, Groovy</i>	<i>VBScript</i>
Навички програмування	Переважно навички вимагаються для інструментів інтеграції.	Немає вимог. Рекомендовано переважно для тестових скриптів.	Немає вимог. Рекомендовано переважно для тестових скриптів.
Легкість використання	Вимоги навички встановлення та використання	Легкий у встановленні та використанні	Складний у встановленні. Потрібна підготовка для використання інструменту.
Доступність / Ціна	Відкритий ресурс	На рік – 759 доларів	На рік 3200 - доларів
Рейтинг	4.4 / 501	4.4 / 651	4.3 / 103



### 2.3. Характеристика інструменту *Selenium*

*Selenium* – це набір інструментів, призначених для автоматизації веб-браузерів на різних платформах. *Selenium* може автоматизувати безліч різноманітних браузерів на різних платформах, використовуючи різні мови програмування і інтегруючись з різними тестовими фреймворками.

*Selenium Webdriver* – популярний інструмент для управління реальним браузером, який можна використовувати як для автоматизації тестування веб-додатків, так і для виконання інших рутинних завдань, пов'язані з роботою у вебі.

Крім того, *Webdriver* – проєкт з відкритим вихідним кодом, підтримує безліч мов програмування та має велике співтовариство користувачів. [17]

Це, мабуть, найпопулярніший інструмент автоматизації тестів і здебільшого підходить для інтерфейсу користувача автоматизації *web*-додатків. Як і всі засоби автоматизації інтерфейсу користувача, *Selenium* дозволяє нам імітувати дії миші та клавіатури від імені користувача та отримувати дані, які відображаються.

*Selenium* має кілька важливих переваг, які роблять його таким популярним:

- Це відкритий код, тому безкоштовний.
- Він підтримує дуже широкий діапазон браузерів.
- Він доступний багатьма мовами програмування.
- Допомогає створювати складні та вдосконалені сценарії автоматизації.
- Основа для більшості інших засобів тестування програмного забезпечення.
- Підтримує паралельне виконання тесту, тим самим скорочуючи час виконання тесту.

Основним недоліком *Selenium* є те, що він розроблений переважно для браузерів і має лише обмежену підтримку інших технологій інтерфейсу через зовнішні розширення. До того ж він в основному призначений для використання з кодом.

Для роботи з *Webdriver* необхідно 3 основні програмні компоненти:

1. Браузер, роботу якого користувач хоче автоматизувати. Це реальний браузер певної версії, встановлений на певній ОС і має свої налаштування.

2. Для управління браузером абсолютно необхідний *driver* браузера. *Driver* насправді є *web*-сервером, який запускає браузер і відправляє йому команди, а також закриває його. У кожного браузера свій *driver*. Пов'язано це з тим, що у кожного браузера свої відмінні команди управління і реалізовані вони по-своєму. Знайти список доступних драйверів і посилання для скачування можна на офіційному сайті *Selenium* проекту.

3. Скрипт / тест, який містить набір команд певною мовою програмування для драйвера браузера. Такі скрипти використовують *Selenium Webdriver bindings* (готові бібліотеки), які доступні користувачам на *Selenium Webdriver* різних мовах.[17]

Ця частина спілкується за допомогою драйвера браузера за допомогою виділеного дротового протоколу *JSON*.

Драйвер браузера отримує запити від мовної прив'язки та викликає файл відповідної операції в браузері. Кожен тип браузера має власний драйвер.

Оскільки всі драйвери «розуміють» один і той же дротовий протокол *JSON*, той самий тест може використовуватися з іншим драйвером та відповідним браузером (рис. 2.1).

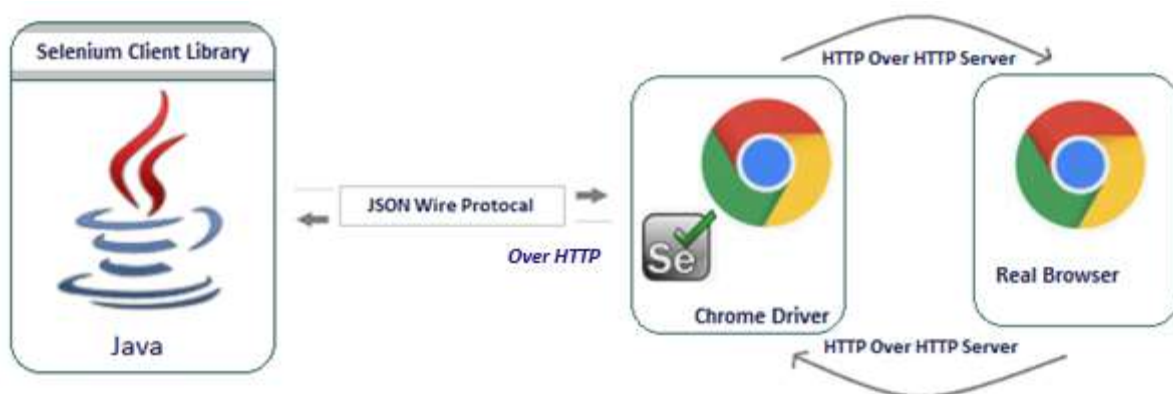


Рис. 2.1. *Selenium WebDriver Automation Framework*.

## ***Selenium IDE***

***Selenium*** також пропонує спеціальний плагін для *Chrome* і *Firefox*, який називається *Selenium-IDE*. Цей плагін дозволяє записувати тестові кейси, а також базове управління та редагування цих тестових кейсів без написання коду. Це також дозволяє експортувати тести в код на *Ruby*, *Java* або *C #*, використовуючи різноманітні популярні фреймворки тестування. [17]

## ***Selenium Grid***

Якщо необхідно масштабуватись, розподіляючи та запускаючи тести на декількох машинах та керуючи кількома середовищами з центральної точки, полегшуючи запуск тестів на широкій комбінації браузерів / ОС, тоді можна використовувати *Selenium Grid*. [17]

*Selenium* - це загальноприйнята назва, коли мова йде про тестування автоматизації. Вважається галузевим стандартом для тестування автоматизованого інтерфейсу *web*-додатків. Зокрема, 54% користувачів використовували *Selenium* як інструмент для автоматизації тестування, згідно з Тестом звіту про автоматизацію автоматизації в 2020 році.

Для розробників та тестувальників, які мають досвід та навички програмування та написання скриптів, *Selenium* пропонує гнучкість, яку не спостерігають у багатьох інших інструментах та рамках автоматизації тестування. Користувачі можуть писати тестові сценарії різними мовами (такими як *Java*, *Groovy*, *Python*, *C #*, *PHP*, *Ruby* та *Perl*), які працюють у декількох системних середовищах (*Windows*, *Mac*, *Linux*) та браузерах (*Chrome*, *Firefox*, *IE* та інших).[17]

Щоб ефективно використовувати *Selenium*, користувачі повинні володіти передовими навичками програмування і їм потрібно витратити значний час на створення систем автоматизації та бібліотек, необхідних для автоматизації. Це головний недолік *Selenium*, який розглядається в інших інструментах, побудованих для безкоштовної автоматизації тестів, таких як *Katalon Studio*.

## 2.4. Характеристика інструменту *Katalon Studio*

*Katalon Studio* – це інструмент автоматизації з відкритим кодом, який підтримує як веб-середовища, так і середовища розробки мобільних додатків.

*Katalon Studio* – це потужне комплексне рішення для автоматизації тестування *API*, тестування веб-програм, мобільних пристроїв та настільних програм. Він також має багатий набір функцій для цих типів тестування та підтримує декілька платформ, включаючи *Windows*, *macOS* та *Linux*. [10]

Це один з найкращих інструментів для тестування автоматизації, який працює зверху на *Selenium* та *Appium*, тим самим покращуючи ці фреймворки такими функціями, як об'єктний шпигун, зручна *IDE*, сховище об'єктів та плагін браузера. Він може бути інтегрований з безліччю інших інструментів, таких як *JIRA*, *qTest*, *Kobiton*, *Git*, *Slack* тощо.

Платформа виділяється своїми численними цілями та простотою використання, враховуючи, що вона може створювати та використовувати повторно тестові сценарії інтерфейсу користувача, не вимагаючи жодного коду.

Інструмент використовує *Groovy* як мову сценаріїв та підтримує зовнішню бібліотеку *Java*. *Katalon* дозволяє повторно використовувати сценарії *Selenium*, написані на *Java*, і використовувати безпосередньо в інструменті. Він безперебійно працює з системами безперервної інтеграції, такими як *Jenkins*, *Bamboo* та *TeamCity*.

Основні моменти інструменту включають:

- Повний набір функцій для тестової автоматизації *API* / веб-сервісів, Інтернету та мобільних додатків.
- Підтримує *SOAP* та *RESTful* для тестування *API* та служб.
- Сотні вбудованих ключових слів для створення тестових кейсів.
- Підтримка *BDD Cucumber* для вираження сценарію тестування природними мовами.
- Може використовуватися як для автоматизованих, так і для дослідницьких випробувань.

- Можливість тестування можна розширити за допомогою плагінів на *Katalon Store*.

- Перегляд звітів про якість на *Katalon TestOps*, тестовій платформі для оркестровки для команд *Agile*.

Переваги такого інструменту:

- Універсальний, оскільки він працює на *Windows*, *macOS* та *Linux*.
- Сотні вбудованих ключових конструкцій для створення тестових кейсів.
- Мінімальні навички програмування, необхідні для використання цього інструменту.

- Підходить як для досвідчених тестувальників, так і для новачків.
- Відсутня складність установки і визначення структури як, наприклад, в *Selenium*.

- Виконання декількох тестів одночасно.
- Інтегрується з інструментами менеджменту вимог і тестування такими як *JIRA* і *qTest*.

Недоліки:

- На відміну від інструментів, таких як *Selenium*, створення скриптів обмежено лише *Java* та *Groovy*.

- На даний момент підтримка розподіленого тестування відсутня.
- Він не може автоматизувати настільні програми, на відміну від деяких ліцензованих інструментів, таких як *UFT* та *TestComplete*.

Крім того, *Katalon* підтримує паралельне та послідовне виконання та може виконувати віддалене та локальне тестування.

## 2.5. Характеристика інструменту *UFT*

*QTP* – це інструмент автоматизованого функціонального тестування, який допомагає тестувальникам виконувати автоматизовані тести з метою виявлення будь-яких помилок, дефектів, що суперечать очікуваним результатам програми, що

тестується. Повна форма *QTP* – це *QuickTest Professional*, тоді як *UFT* означає уніфіковане функціональне тестування.

*UFT One* (раніше відомий як *UFT*) – популярний комерційний інструмент для тестування веб-програм, настільних ПК, мобільних пристроїв та додатків *RPA*. Він був розширений, включаючи хороший набір можливостей для тестування *API*. Підтримуючи кілька платформ для цільового тестованого додатка (*AUT*), *UFT One* пропонує зручний вибір для тестування *AUT*, який працює на настільних комп'ютерах, в Інтернеті та на мобільних пристроях.

Він може автоматизувати Інтернет, Робочий стіл, *SAP*, *Java*, *Oracle*, *Mobile* та *Visual Basic*, серед інших програм. Список середовищ розробки, які він може автоматизувати, величезний, і його можна використовувати для різних видів тестування програмного забезпечення. Він виконує функціональне та регресійне тестування через користувальницький інтерфейс, такий як власний графічний інтерфейс або веб-інтерфейс. [20]

*UFT* використовує *VBScript* як мову сценаріїв. Інструмент тісно інтегрований з *HP ALM* (Засіб управління тестами) та *HP LoadRunner* (Інструмент тестування продуктивності) (рис. 2.2).

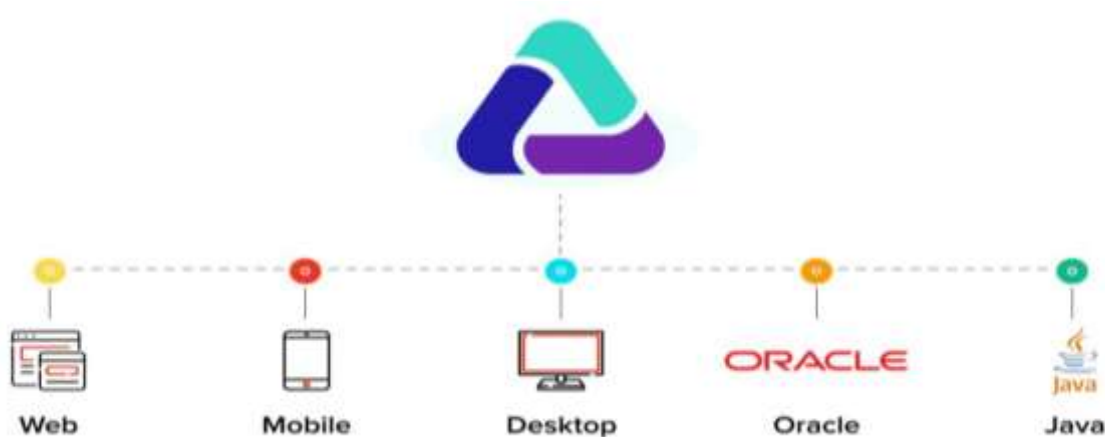


Рис. 2.2. *HPE Unified Functional Testing (UFT)*.

Кілька основних моментів інструменту:

- Створить більш агностичний тест на платформі із запропонованими кроками об'єкта в *AI Transformation Assistant*.

- Використовує *ParallelRunner* для паралельного запуску декількох тестів *API* та графічного інтерфейсу.
- Відстежує та повідомляє про стан *Wi-Fi*, тепловий стан та доступні метрики дискового простору в налаштуваннях *Record & Run*.
- Підтримка автоматичних контрольних точок на сторінках для браузерів *Firefox* або *Chrome*.
- Використовує *VBScript* як мову сценаріїв.
- Підтримує тестування на основі даних.
- Пропонує крос-браузерну та багатоплатформну сумісність.

Недоліки:

- *UFT* надзвичайно дорогий, ніж інші засоби автоматизації. В основному його ліцензія коштує дорожче, а також додаткові збори за будь-які надбудови.
- Незважаючи на те, що в *UFT* час сценаріїв менше, час виконання помірно більший у порівнянні з іншими інструментами.
- *UFT* працює не у всіх версіях браузерів.
- Щоб отримати будь-яку онлайн-підтримку від *HP*, користувачеві потрібно поновити ліцензію.
- *UFT* не підтримує багатопоточність.
- Пов'язування або інтеграція *UFT* з іншими інструментами є дуже дорогим.
- І міграція даних в *UFT* також є перешкодою.[1]

Ваш вибір засобів тестування повинен не тільки відповідати вашим поточним потребам, але також зосередитись на потенційних тенденціях та вдосконаленнях. Пристойний інструмент повинен підтримувати базову оптимізацію, автоматизацію тестування та формування даних, розумніші рішення та аналітику.

## **Висновки до розділу**

Наразі у наш час є багато засобів тестування автоматизації. Кожен інструмент цікавий, унікальний і має свої переваги та недоліки. Великий вибір інструменту для автоматизовано тестування ускладнює вибір найбільш необхідного для проєкту, і часто на кінцевому етапі тестувальники отримують інструменти, які не відповідають вимогам проєкту. Було описано інструменти для автоматизованого тестування. Розглянули найголовніші критерії вибору інструменту. Порівняли 3 найбільш популярні інструменти для автоматизації, оцінили їхні параметри, переваги та недоліки роботи з інструментом. Отже, вибір правильного інструменту для проєкту дуже важливий.



## РОЗДІЛ 3

### РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

#### 3.1. Опис *Web*-додатку

На даному етапі необхідно описати сам *web*-додаток, на основі якого буде розроблятися тест-план, на основі документації. Етапи функціонального тестування написання тест-кейсів, робота перевірки функціональності самого сайту та його компонентів (форм, контакти, зв'язок, технічна підтримка)

Для автоматизованого тестування було обрано веб-додаток інтернет магазину одягу: <https://www.sinsay.com/ua/ru/>

*Sinsay* – 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.

В процесі тестування сайту *Sinsay* буде використано *ad-hoc* тестування через відсутність специфікації, а також через обмеженість ресурсів на формалізацію тестів.

Планується чотири етапи проведення процесу тестування:

- перший етап полягає в складанні тест-плану і чек-листа, а також часткового прогону функціональних тестів;
- другий етап буде складатися з тестування верстки сайту;
- на третьому етапі буде проведений прогін авто тестів;
- четвертий етап полягає в тестуванні зручності використання продукту з описом пропозицій щодо поліпшення.

Таким чином, досягається максимальна деталізація глибини тестування, що, в свою чергу, дозволяє більш точно визначити витрачені ресурси, а також дозволяє розробникам проєкту виправляти дефекти на ранніх етапах.

*ОС*, яка буде використовуватися під час тестування:

- *Windows 10*

Браузер, який буде використовуватися під час тестування:

- *Google Chrome 90*

Тестування безпеки і стрес-тестування не проводиться.

### **3.2. Необхідні інструменти для створення автотестів**

У другому розділі було розглянуто різні інструменти, за допомогою яких можливо відобразити автотести нашого веб-додатку.

Було обрано один інструмент із перелічених – це *Selenium WebDriver*.

Це, мабуть, найпопулярніший інструмент автоматизації тестів і здебільшого підходить для інтерфейсу користувача автоматизації веб-додатків. Як і всі засоби автоматизації інтерфейсу користувача, *Selenium* дозволяє нам імітувати дії миші та клавіатури від імені користувача та отримувати дані, які відображаються.

*Selenium WebDriver* – це простий *API*, який може допомогти в автоматизації браузера. Однак набагато більше необхідний при використанні його для тестування та побудови тестової основи.

Нам знадобиться інтегроване середовище розробки (*IDE*) або редактор коду для створення нового проєкту *Java* та додавання *Selenium WebDriver* та інші залежності для побудови основи тестування.

*Selenium WebDriver* встановлюємо із офіційного джерела.

<https://www.selenium.dev/>

Також будемо використовувати *IntelliJ IDEA*.

*IntelliJ IDEA* аналізує код, шукаючи зв'язки між символами у всіх файлах проєкту та мовах. Використовуючи цю інформацію, він надає допомогу в глибокому кодуванні, швидку навігацію, розумний аналіз помилок.

<https://www.jetbrains.com/ru-ru/idea/>

У світі *Java Eclipse* є широко використовуваною *IDE*, а також *IntelliJ IDEA* та *NetBeans*. *Eclipse* забезпечує багатофункціональне середовище для розробки тесту *Selenium WebDriver*. Поряд з *Eclipse*, *Apache Maven* надає підтримку для управління всім життєвим циклом тесту проєкту. *Maven* використовується для визначення структури проєкту, залежностей, побудови та управління тестами.

Ми будемо використовувати *Maven* для побудови тестового середовища *Selenium WebDriver*. Ще однією важливою перевагою використання *Maven* є те, що можна отримати всі *Selenium* файли бібліотеки та їх залежності шляхом налаштування файлу *pom.xml*. *Maven* автоматично завантажує необхідні файли зі сховища під час створення проєкту.

Необхідно також завантажити *Maven*.

<https://maven.apache.org/>

### 3.3. Вибір мови програмування

При виборі мови сценаріїв слід враховувати наступні фактори.

- Сучасна мова, що використовується для розвитку.
- Підтримка розробників мови програмування локально. Тестери повинні розглянути можливість використання мови, за якою стежать розробники.
- Для тих, хто новачок у процесі, краще використовувати *Python*, *Ruby* або будь-які інші мови, зручні для виконання сценаріїв. Він вимагає менше коду і може бути написаний швидко, без зайвих клопотів.
- Хоча *Java* є найпоширенішою мовою, її синтаксис дещо складний для реалізації в скриптах.

Ми реалізуємо наші автотести за допомогою *Java*.

*Java* — це мова програмування загального призначення, яка належить корпорації *Oracle*. *Java* побудована на принципах об'єктно-орієнтованого програмування. Мова дотримується принципу *WORA*, який приносить багато переваг між платформами.

Багато великих корпорацій використовують *Java* для обслуговування своїх внутрішніх систем. Існує більше 3 мільярдів пристроїв, на яких запущено додатки, побудовані за допомогою *Java*.

*Selenium* є корисним інструментом, оскільки він є не лише відкритим кодом, а й портативною програмою тестування програмного забезпечення для веб-додатків, що підтримують різні мови, такі як *Java*, *C #*, *Ruby*, *Python*. Вибір правильної мови залежить від програми, що тестується, спільноти, що підтримує, доступних систем автоматизації тестів, зручності використання, елегантності та, звичайно, безперебійної інтеграції збірки. [18]

*Java* – популярна мова програмування. Відповідно до *StackOverflow*, це третя за популярністю внутрішня технологія після *JavaScript* та *SQL* (рис. 3.1).

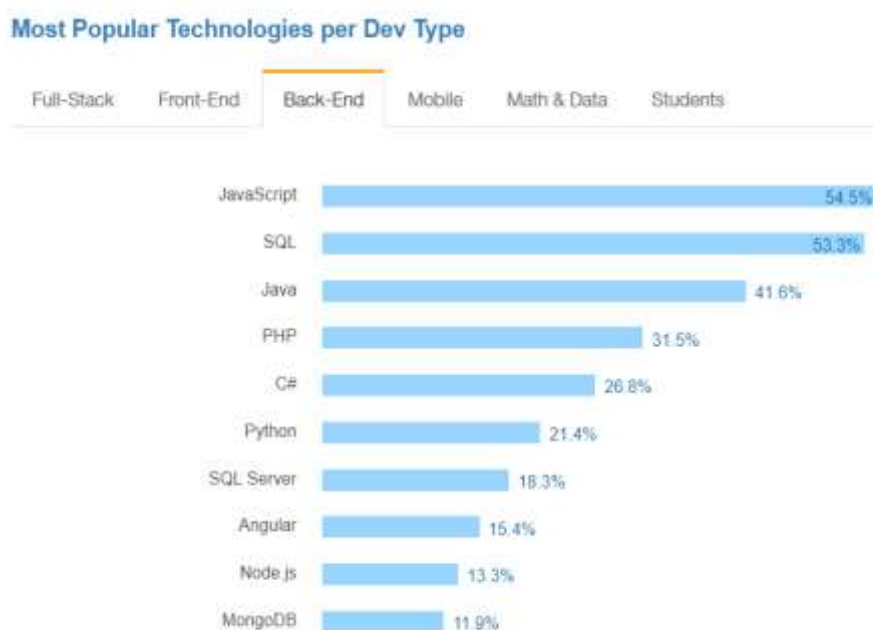


Рис. 3.1. Рейтинг найбільш популярних технологій.

Наступний крок встановлення *Java SE 15*

<https://www.oracle.com/java/>

### 3.4. Відображення процесу проходження документації

Після встановлення усіх необхідних додатків, ми можемо переходити до опису документації.

Перший етап полягає в складанні тест-плану і чек-листа, а також часткового прогону функціональних тестів. Це було зазначено вище в етапах, тому почнемо з цього.

Тест- план для інтернет магазину (таблиця 3.1).

Таблиця 3.1

Таблиця інформації створення тест-плану

Дата	Версія	Вид змін	Автор
04.06.2021	1.0	Створення	Белозьорова Д.О.

В описі тест-плану будуть присутні такі пункти:

- Вступ
- Мета
- Вихідні дані
- Мета тестування
- Умови тестування
- Стратегія процесу тестування
- Типи тестування
- Функціональне тестування
- Тестування кросбраузерності
- Регресивне тестування і перевірка вирішених дефектів
- Тестування дизайну
- План робіт
- Кінцеві результати
- Висновки

Метою складання даного тест-плану є опис процесу тестування сайту *Sinsay* (повна адреса <https://www.sinsay.com/ua/ru/>). Опис дозволяє отримати уявлення про планові роботи з тестування проекту.

## 1.2 Вихідні дані

*Sinsay* – 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, ділитися описом товарів та ін.

## 1.3 Мета тестування

Метою тестування сайту *Sinsay* є перевірка коректної роботи всіх його функціональних можливостей з типовими сценаріями його використання.

Результатом процесу тестування будуть наступні матеріали:

- підсумок тестування відносно загального стану, що дає розробникам і менеджерам даного продукту картину відносно коректності роботи сайту;
- звіт про результати тестування поточного покриття, типові сценарії використання;
- задокументовані баги в баг-трекері замовника.

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

Умови для тестування.

*Web*-сайт повинен задовольняти потреби користувача в активностях, пов'язаних з переглядом каталогів товарів, пошуком товарів, переглядом опису товарів, додаванням в кошик / вибраного, оформленням заявки, публікування в соц. мережі.

Стратегія процесу тестування.

Наведений нижче план тестування є формальним, так як для побудови розгорнутого плану необхідно розуміння поточного стану проекту.

В процесі тестування сайту *Sinsay* буде використано *ad-hoc* тестування через відсутність специфікації, а також через обмеженість ресурсів на формалізацію тестів.

Планується чотири етапи проведення процесу тестування:

- перший етап полягає в складанні тест-плану і чек-листа, а також часткового прогону функціональних тестів;
- другий етап буде складатися з тестування верстки сайту;
- на третьому етапі буде проведений прогін авто тестів;
- четвертий етап полягає в тестуванні зручності використання продукту з описом пропозицій щодо поліпшення.

Таким чином, досягається максимальна деталізація глибини тестування, що, в свою чергу, дозволяє більш точно визначити витрачені ресурси, а також дозволяє розробникам проєкту виправляти дефекти на ранніх етапах.

ОС, яка буде використовуватися під час тестування:

- *Windows 10*

Браузер, який буде використовуватися під час тестування:

- *Google Chrome 90*

Тестування безпеки і стрес-тестування не проводиться з причини необхідності тестування.

Типи тестування.

## **1. Функціональне тестування**

Мета: виявлення функціональних помилок, невідповідностей ТЗ і очікуванням користувача шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Опис процесу:

- Реєстрація/Авторизація
- Вхід покупця на сайт
- Перегляд каталогу
- Розрахунок товару
- Особистий кабінет
- Зворотній зв'язок
- Пошук
- Фотогалереї

- Банери
- Коментарі

## 2. Тестування кросбраузерності

Мета: перевірити коректну роботу і дизайн проєкту, будемо перевіряти тільки на одному браузері.

- *Google Chrome 90*

## 3. Регресивне тестування і перевірка вирішених дефектів

Мета: перевірка змін, зроблених на сайті для того, щоб впевнитися, що нова версія не має помилок в уже протестованих частинах сайту.

Так як ми тестуємо вже повністю функціональний сайт, в цьому немає потреби. Перевіримо його роботоздатність.

## 4. План робіт (таблиця 3.2).

Таблиця 3.2

Таблиця плану робіт

Задача	Об'єм роботи	Дата початку	Дата закінчення
Складання тест-плану	5 годин	04.06.2021	04.06.2021
Виконання тестування	12 годин	04.06.2021	05.06.2021
Аналіз тестування	2 години	05.06.2021	06.06.2021
Підведення підсумків	3 години	06.06.2021	06.06.2021

## Кінцеві результати

### Висновок

Кінцевим підсумком проведення тестування повинен стати оформлений кінцевий результат процесу тестування з описаними дефектами, а також рекомендаціями про покращення продукту з точки зору кінцевого користувача.



Тест-кейс - набір вхідних даних, умов виконання та очікуваних результатів, розроблений з метою перевірки тієї чи іншої властивості або поведінки програмного засобу.

Під тест-кейсом також може розумітися відповідний документ, який представляє формальний запис тест-кейса (рис. 3.2.).

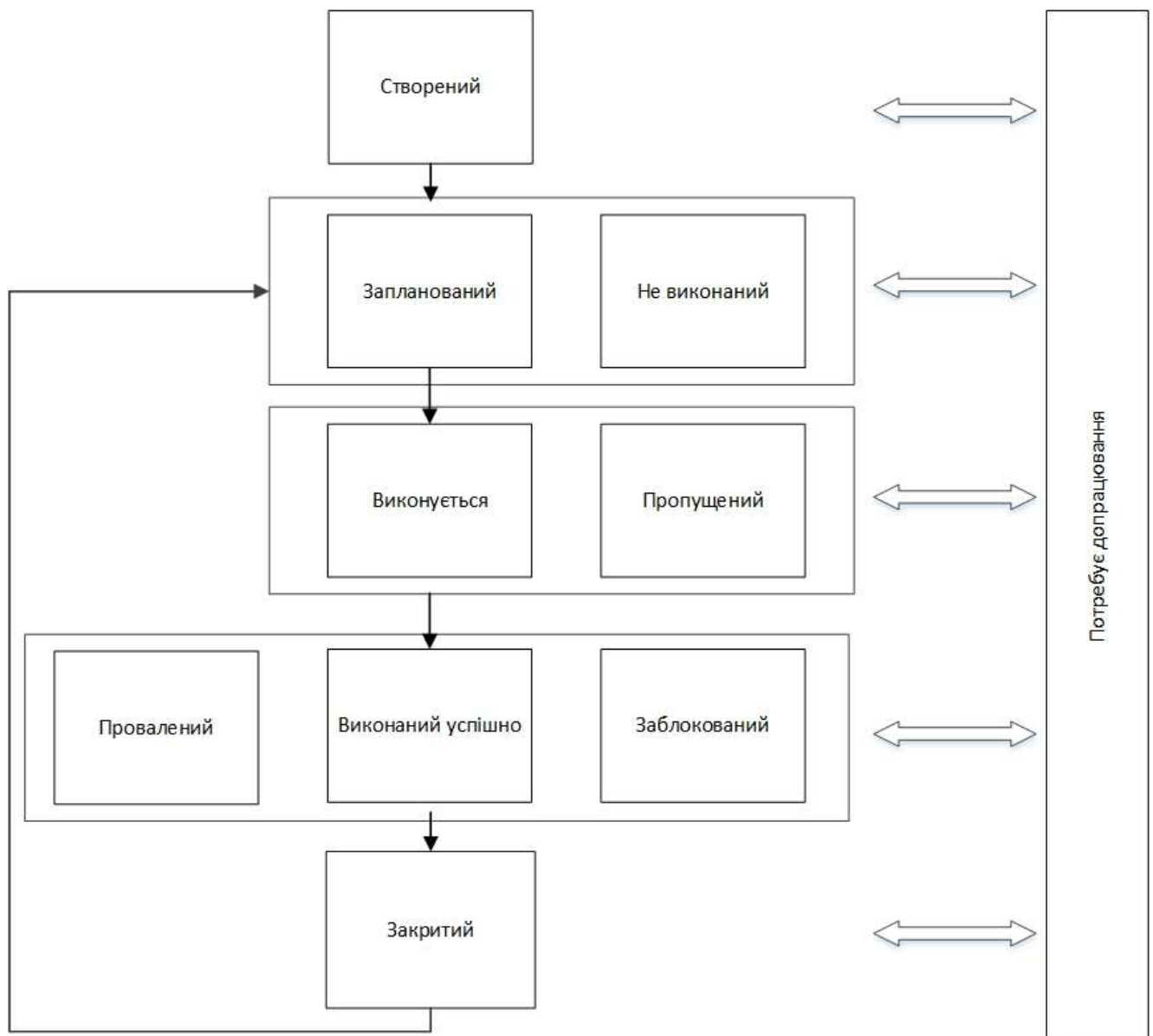


Рис. 3.2. Життєвий цикл тест-кейсу.

Для нашого сайту створимо декілька тест-кейсів з описом кроків тестування для автотестів. Це будуть тільки деякі найбільш використовувані кроки. Розглянемо як позитивні так і негативні сценарії.

### 3.5. Автоматизоване тестування системи веб-додатку

#### 1. Створення проєкту.

У додатку *IntelliJ IDEA* створюємо новий проєкт за допомогою *Maven* і обираємо версію *Java*. Проєкт створили під назвою «*Test*» (рис. 3.3.).

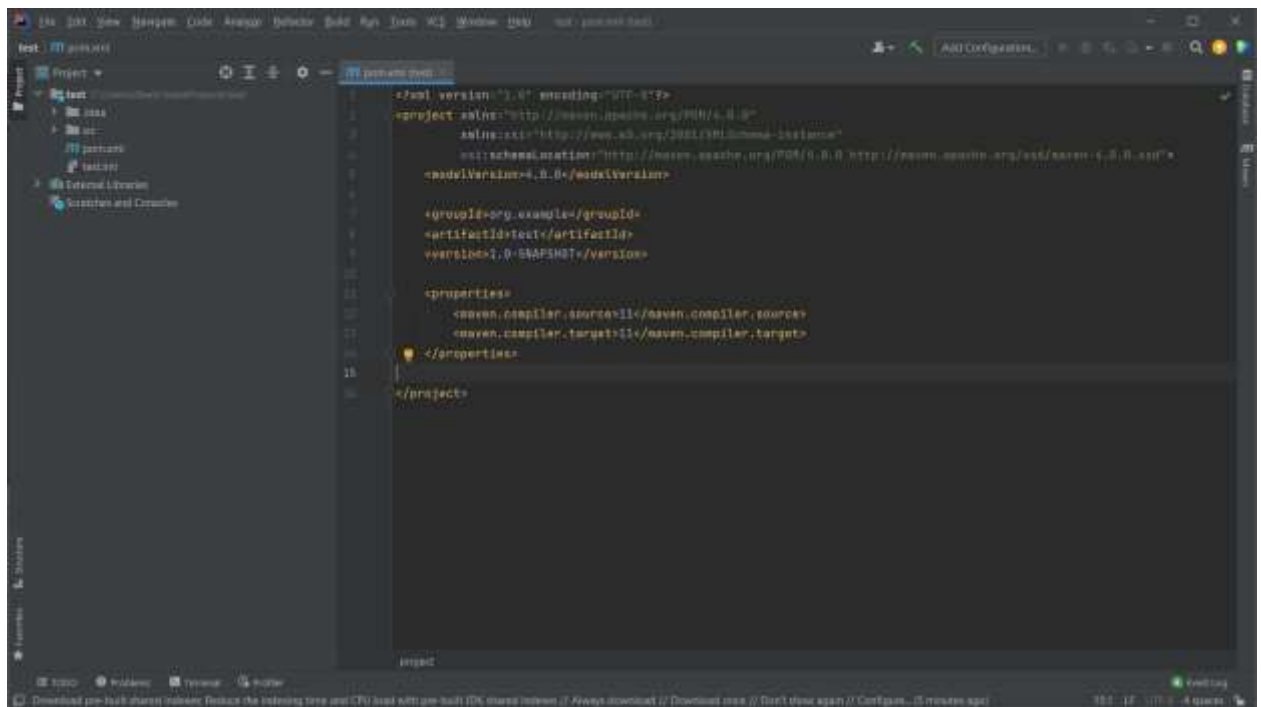


Рис. 3.3. Головний екран проєкту.

Так виглядає наша початкова сторінка створеного проєкту «*Test*».

*GroupId* і *Artifactid* – ідентифікатори проєкту в *Maven*. Існують певні правила заповнення цих пунктів:

- *GroupId* - назва організації або підрозділу займаються розробкою проєкту. У цьому пункті діє теж правило як і в іменуванні пакетів *Java*: доменне ім'я організації записане задом наперед. Якщо у Вас немає свого доменного імені, то можна використовувати свій е-мейл, наприклад *com.email.email*;

- *Artifactid* – назва проєкту;
- *Version* – версія проєкту.

Необхідно налаштувати наш проєкт та підключити драйвер *Google Chrome*.

Спочатку нам потрібно встановити прив'язки *Selenium* для проєкту автоматизації.

Встановлення бібліотек *Selenium* для *Java* можна здійснити за допомогою *Maven*.

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>LATEST</version>
</dependency>
```

Щоб створити екземпляр сеансу *Chrome*, ми зробимо наступне:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
System.setProperty(«webdriver.chrome.driver»,
«C:\\Program Files\\chromedriver_win32\\chromedriver.exe»);
//слід встановити залежність, що визначає шлях до chromedriver
WebDriver driver = new ChromeDriver();
//для запуску браузера необхідно створити об'єкт драйвера
```

Також використаємо методи відкриття нашої сторінки <https://www.sinsay.com/ua/ru/> на повний екран.

```
driver.manage().window().maximize();
```

Цей приклад викликає *get* метод *driver* змінної для переходу до веб-сторінки за адресою <https://www.sinsay.com/ua/ru/>, передаючи значення аргументу рядка для її *URL*-адреси.

```
driver.get(«https://www.sinsay.com/ua/ru/»);
```

Щоб завершити тест, викличемо *quit* метод на екземплярі *WebDriver* інтерфейсу, наприклад, на *driver* змінній.

```
driver.quit();
```

На цьому налаштування нашого проєкту автоматизації завершено. Далі переходимо до написання самих тест-кейсів та створення автотестів до них.

Головна сторінка нашого веб-додатку ( рис. 3.4.).

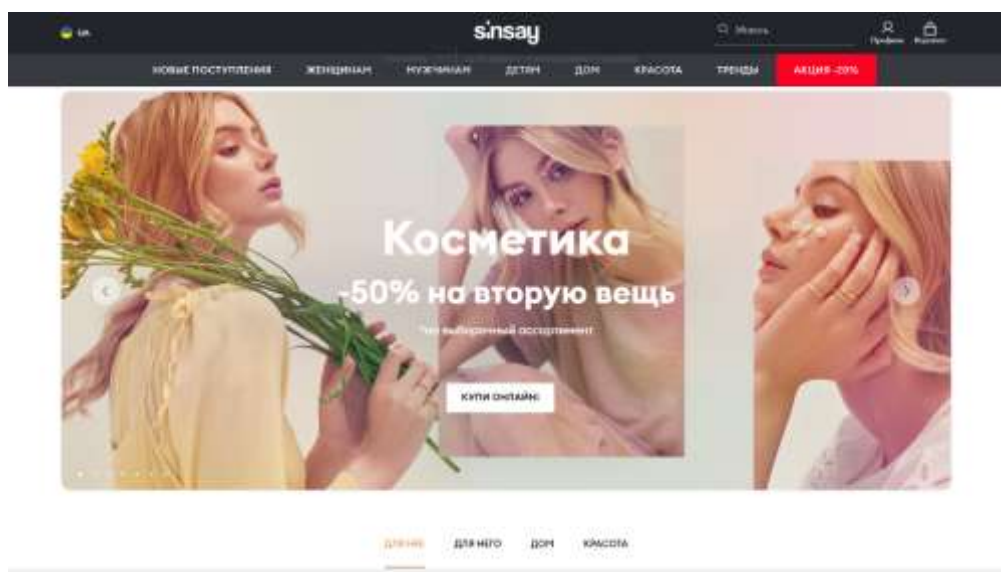


Рис. 3.4. Головна сторінка веб-додатку.

Опишемо позитивний сценарій тестування у вигляді тест-кейсу.

### Опис позитивного тест-кейсу №1

**Опис тесту: ціль, сценарій і вихідний стан програми.**

Вхід в систему користувача.

**Передумови:**

1. Користувач зареєстрований.

Кроки	Очікуваний результат
Відкрити сайт « <a href="https://www.sinsay.com/ua/ru/">https://www.sinsay.com/ua/ru/</a> »	Відкривається головна сторінка.
Натиснути на кнопку «Профиль»	Відкривається сторінка авторизації.
Ввести в поле «Адрес электронной почты» дійсний email « <i>kiraparizeva@gmail.com</i> »	Введена адреса прийнята, поле не підсвічується червоною рамкою.
Ввести в поле «Пароль» дійсний пароль « <i>kida1999</i> »	Введений пароль співпадає.
Натиснути на кнопку «Войти»	Відкривається головна сторінка авторизованого користувача.

Лістинг виконання автотесту (додаток А).



Рис. 3.5. Результат успішного збирання тесту в програмі *IntelliJ IDEA*

### Опис негативного тест-кейсу №1

**Опис тесту: ціль, сценарій і вихідний стан програми.**

Повідомлення про помилку під час входу в систему користувача.

**Передумови:**

1. Користувач зареєстрований.

Кроки	Очікуваний результат
Відкрити сайт « <a href="https://www.sinsay.com/ua/ru/">https://www.sinsay.com/ua/ru/</a> »	Відкривається головна сторінка.
Натиснути на кнопку «Профіль»	Відкривається сторінка авторизації.
Ввести в поле «Адрес электронной почты» дійсний email « <i>kiraparizeva@gmail.com</i> »	Введена адреса прийнята, поле не підсвічується червоною рамкою.
Ввести в поле «Пароль» не дійсний пароль « <i>kida</i> »	Введений пароль співпадає.
Натиснути на кнопку «Войти»	Відображення помилки «Неверный логин или пароль» під час входу.

Лістинг виконання автотесту (додаток А).

Переглянемо результат у вигляді скріншота екрана під час виконання негативного тест-кейсу з введенням не валідного пароля та виведення повідомлення про помилку (рис. 3.6).

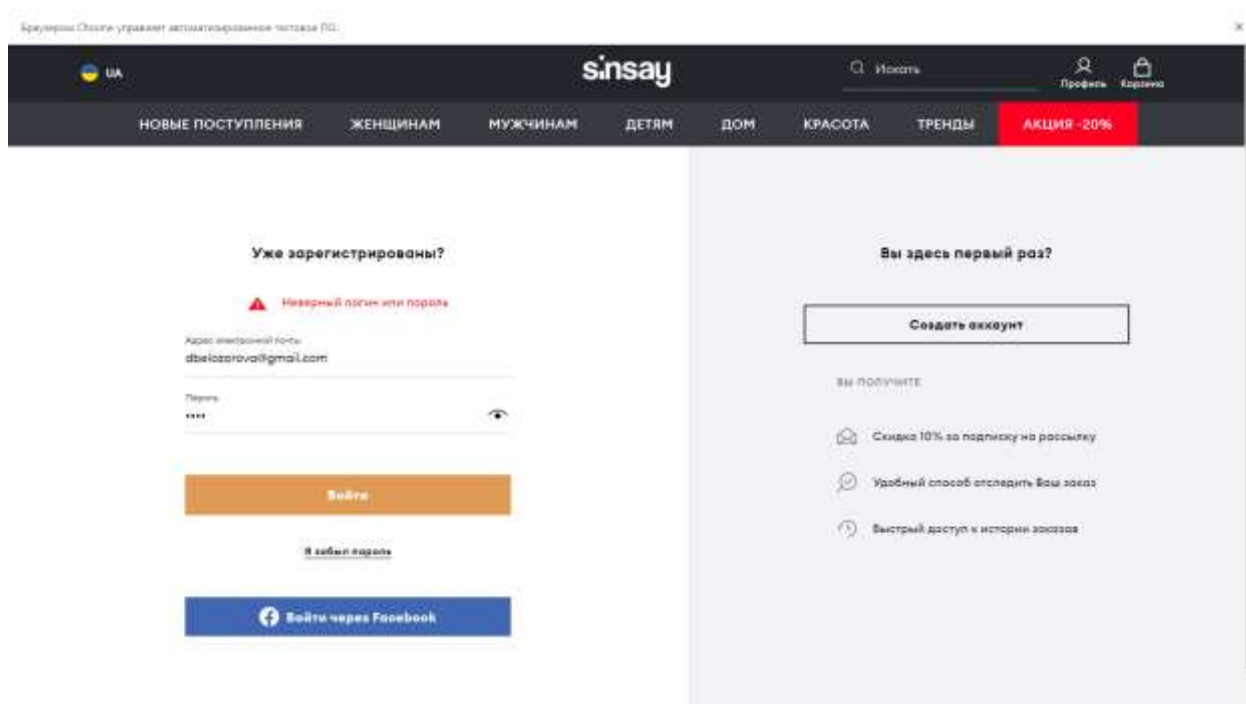


Рис. 3.6. Сторінка авторизації користувача з повідомлення про помилку.

Найбільш використовуваним елементом на сторінці є поле «Пошук» та використання фільтрів. Тому другий тест-кейс буде виконаний на основі поля «Пошук». Введемо у пошук категорію «Платье», оберемо з випадючого списку «Платье женское» та натиснемо *Enter*. Далі використаємо фільтр категорії «Цвета» та виберемо колір «Коричневый» і переглянемо результат.

## Опис позитивного тест-кейсу №2

**Опис тесту: ціль, сценарій і вихідний стан програми.**

Пошук елемента по фільтру.

Кроки	Очікуваний результат
Відкрити сайт « <a href="https://www.sinsay.com/ua/ru/">https://www.sinsay.com/ua/ru/</a> »	Відкривається головна сторінка.
Натиснути на поле «Искать»	Поле виділяється, відображається текстовий курсор.
Ввести в поле «Платье».	Відображають символи у полі вводу.
Обрати з випадаючого списку «Платье женское»	Відображення категорії «Платье женское»
Обрати у фільтрі «Цвета» колір «Коричневый».	Відображення категорії «Платье женское» з кольором «Коричневый».

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

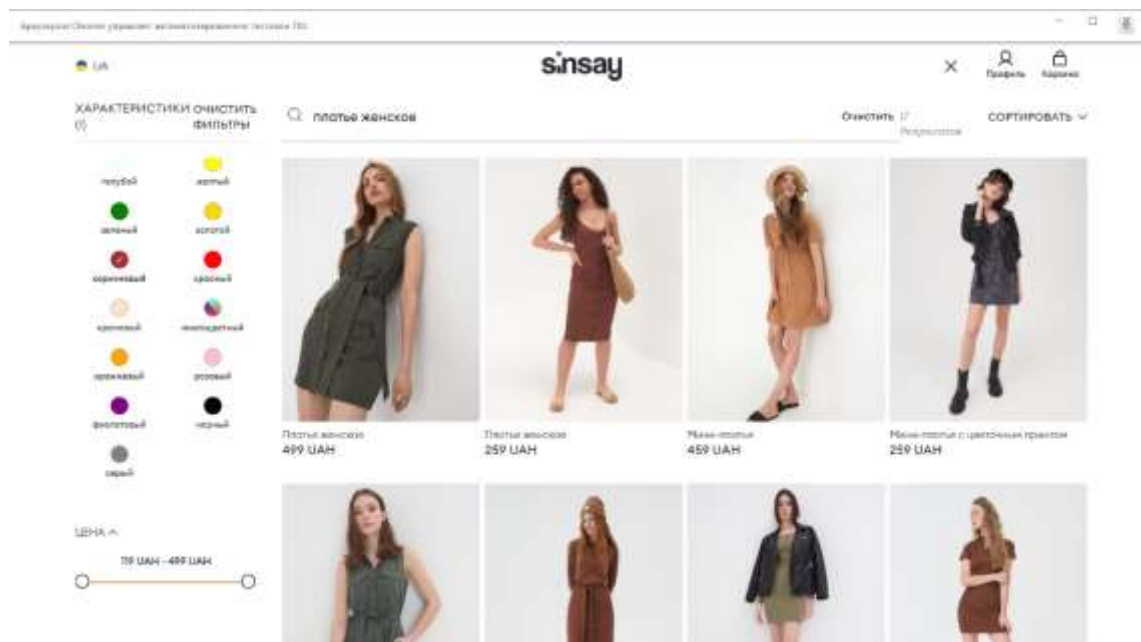


Рис. 3.7. Результат виконання тестового кейсу №2.

Опишемо ще один тест з використання кошика.

Також найбільш використовуваним елементом на сторінці є Корзина, тому третій тест буде виконаний на основі форми Корзина. Введемо у пошук категорію,

обираємо перший товар у категорії та переходимо на нього. Підбираємо колір, розмір і додаємо у корзину. Також для відтворення результату додаємо умову: якщо товар у корзині = 0, то «КОРЗИНА ПУСТАЯ», а якщо > 0, то вивід тексту «ТОВАР БЫЛ ДОБАВЛЕН В КОРЗИНУ». Для відтворення усіх кроків опишемо тест-кейс.

### Опис позитивного тест-кейсу №3

**Опис тесту: ціль, сценарій і вихідний стан програми.**

Додавання та перевірка наявності товару у корзині.

**Передумови:**

1. Користувач зареєстрований.

Кроки	Очікуваний результат
Відкрити сайт « <a href="https://www.sinsay.com/ua/ru/">https://www.sinsay.com/ua/ru/</a> ».	Відкривається головна сторінка.
Натиснути на поле «Искать».	Поле виділяється, відображається текстовий курсор.
Ввести в поле «Платье».	Відображають символи у полі вводу.
Обрати з випадального списку «Платье».	Відображення категорії «Платье» .
Натиснути на перший товар.	Відкривається сторінка обраного товару.
Обрати колір товару.	Колір товару підсвічується лінією.
Обрати розмір товару.	Поле розміру підсвічується чорною рамкою.
Натиснути на кнопку «Добавить».	Товар з'явився у корзині.

Лістинг виконання автотесту наданий в додатку А. Результат корзини після закінчення автотесту (рис. 3.8., рис. 3.9.).



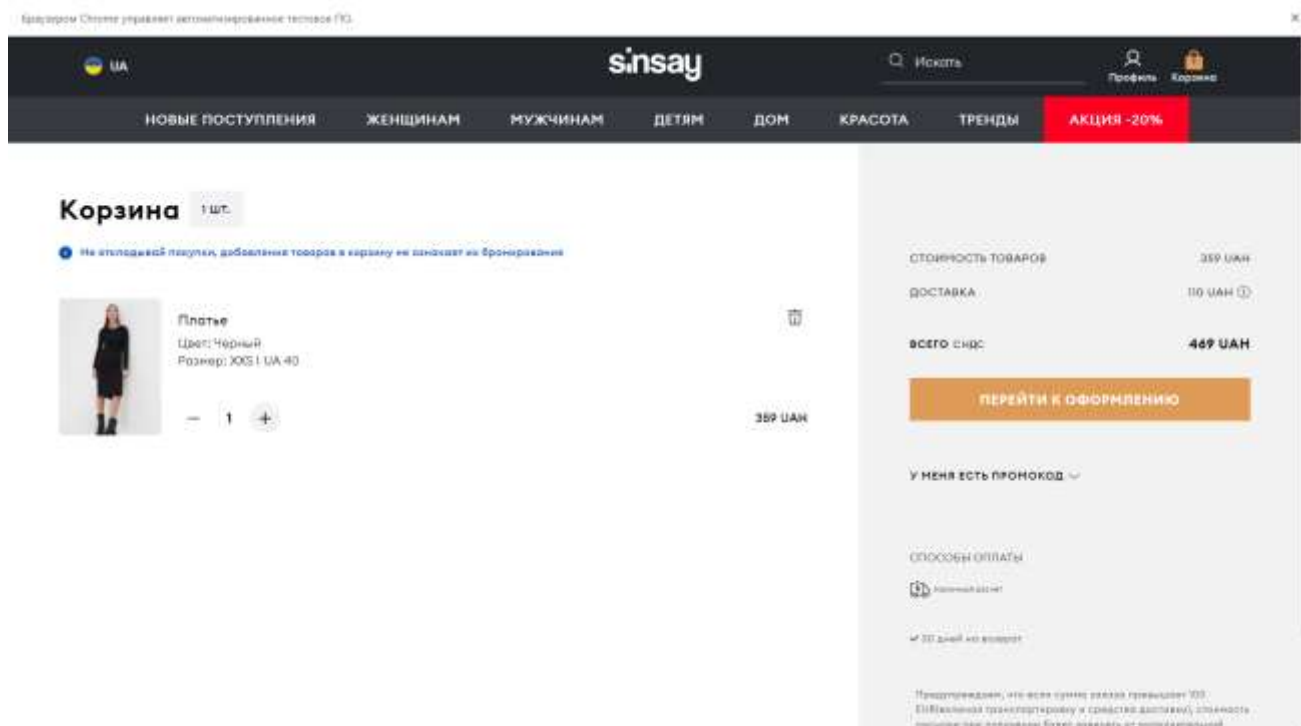


Рис. 3.8. Скріншот корзини після закінчення автотесту.

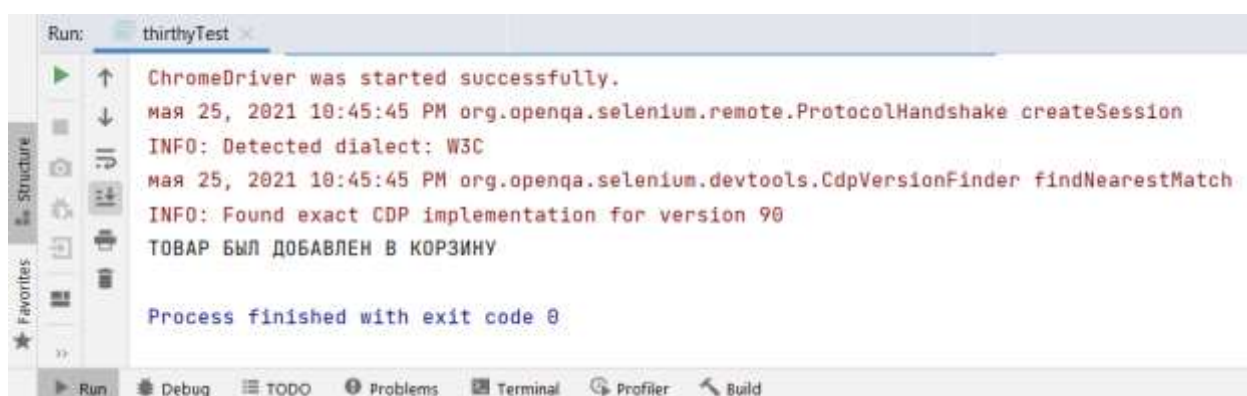


Рис. 3.9. Результат перевірки тесту в Selenium.

### 3.6. Аналіз отриманих результатів

Відтворення усіх автотестів відбувалося на ноутбучі *ASUS X541U*.

#### Параметри системи:

Процесор: *Intel® Core™ i7-3500U CPU @ 2.50GHz/2.59GHz*

Встановлена пам'ять (ОЗП): 16Гб

Тип системи: 64-розрядна.

Операційна система: *Windows 10*.

Після відтворення усіх тестів можемо узагальнити результат у вигляді таблиці, в якій порівняємо критерії ручного та автоматизованого тестування (таблиця 3.3).

Таблиця 3.3

Таблиця порівняння видів тестування

Вид/категорія	Час	Персонал	Ціна	Обслуговування
Ручне	2хв. 1с	1	400\$	-
Автоматизоване	58с.	1	500\$	-

*Manual QA*, без нього, звісно, ми не обійдемося. Працівник спочатку збирає усю необхідну документацію, потім описує увесь цикл розробки, проектування та тестування. Описує тест-план, по необхідності чек-лист та тест-кейси. У нас було описано 4 теста з яких 3 з позитивним сценарієм тестування та 1 з негативним. Далі прогін тестів в ручну. На обраний нами проєкт необхідна мінімальна кількість персоналу 1 (таблиця 3.4.).

Таблиця 3.4

Таблиця часу виконання прогону тестів ручного тестування

№ Тесту	Час виконання
№ 1	32с.
№ 2	32с.
№ 3	29с.
№ 4	28с.

Підрахуємо загальний час проходження Сума = 32+32+29+28=2хв. 1с.

Зарплатню візьмемо за місяць в *Junior QA* по офіційним даним на сайті *dou.ua*. У нас виходить 400\$ (рис. 3.10).

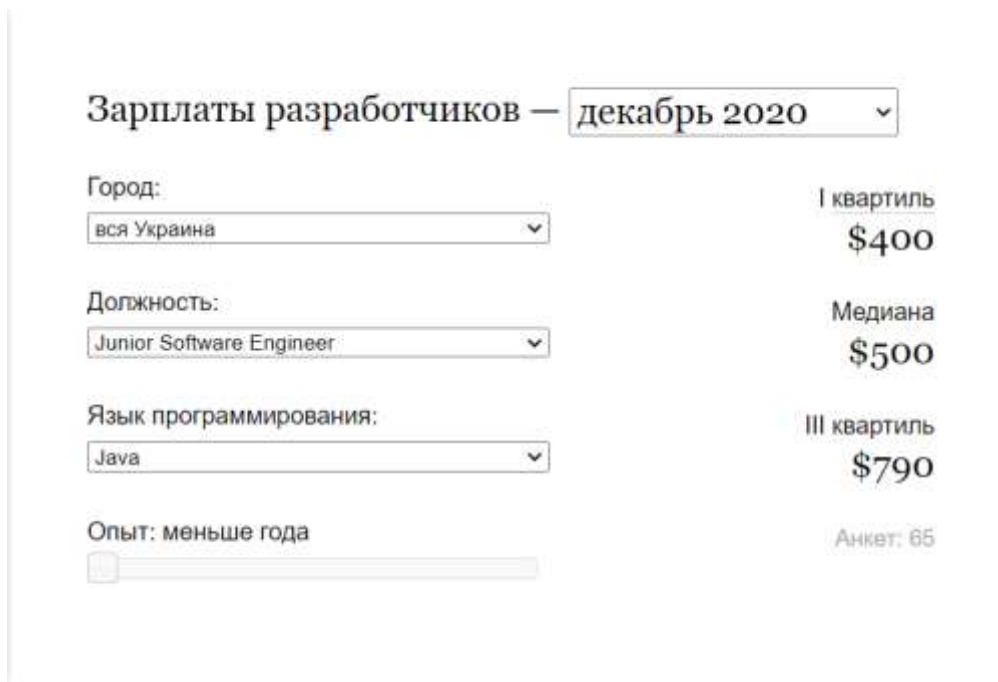


Рис. 3.10. Середня оплата *Junior QA* на старті.

Обслуговування: для ручного тестувальника також є програми, які необхідно використовувати, наприклад (*Charles Proxy*).

*Charles* є проксі-сервером *HTTP* / монітором *HTTP* / зворотним проксі-сервером, який дозволяє розробнику переглядати весь трафік *HTTP* та *SSL / HTTPS* між їх машиною та Інтернетом. Це включає запити, відповіді та заголовки *HTTP* (які містять інформацію про файли *cookie* та кешування).[40].

Програма є у вільному доступі, так що обслуговування опишемо по мінімальному без затрат 0грн.

Тепер перейдемо до автоматизованого тестування.

*Automation QA*, він також робить усю необхідну роботу по документації. Працівник спочатку збирає усю необхідну документацію, потім описує увесь цикл розробки, проєктування та тестування.

На проєкт, обраний нами, необхідна мінімальна кількість персоналу 1.

Час виконання прогону тестів (таблиця 3.5).

Таблиця часу виконання прогону тестів автоматизованого тестування

№ Тесту	Час виконання
№ 1	12с.
№ 2	12с.
№ 3	14с.
№ 4	20с.

Підрахуємо загальний час проходження  $\text{Сума} = 12+12+14+20=58\text{с.}$

Зарплатню візьмемо за місяць в *Junior automation QA* за офіційними даними на сайті *dou.ua*. У нас виходить 500\$

Обслуговування у нас використовується безкоштовне, але для користування ними необхідний підготовлений користувач. Обслуговування опишемо по мінімальному без затрат 0 грн.

Ось так було проведено підсумки і, дивлячись на результати, можна зробити висновки.

1. Візьмемо час, який було використано. Звісно, зрозуміло, що автоматизоване тестування показало себе набагато краще. Тому тут без обговорень.
2. Персонал: для роботи необхідно мінімальну кількість персоналу.
3. Ціна: ціна відрізняється і може варіюватися по-різному, дивлячись від навиків, як технічних так і теоретичних.
4. Обслуговування: використовуючи найкращі програми для тестування, затратність буде з обох боків.

Отже, на основі проведеного дослідження виявлено, що автоматизоване тестування в середовищі *IntelliJ IDEA* з використання мови програмування *Java* вимагає найменшого часу для виконання тестів.

## Висновки до розділу

У третьому розділі було спроектовано систему автотестів, яка складається з трьох позитивних і одного негативного тест-кейсів. Охарактеризували обрані інструменти, прописали тест-план і розробили тест-кейси до кожного тесту. В тест-кейсах було описано ціль, сценарій і вихідний стан програми, передумови, які мають бути включені перед виконання необхідного тест-кейсу. Підключили усі інструменти, встановити прив'язку *Selenium* до проєкту автоматизації. Встановили бібліотеки *Selenium* для *Java* за допомогою *Maven*.

Здійснили реалізацію скриптів для демонстрування роботи системи. Проаналізували роботу та результат виконання.

## ВИСНОВКИ

В ході виконання даної дипломної роботи було проаналізовано переваги та недоліки різних підходів до тестування. Охарактеризували аспекти підходів та методів тестування програмного забезпечення. Розкрили усі теоретичні поняття, що стосуються теми тестування *web*-додатків, і на підставі цих даних була розроблена система тестування.

Вибір правильного інструменту може бути складним завданням. Описали критерії, за якими можна охарактеризувати та в кінцевому результаті обрати необхідний для кожного інструмент. Аналізуючи зібрані дані, було виявлено слабкі і сильні сторони методів та створено рейтинг.

Останнім кроком було проектування самої системи на основі усієї інформації. Встановлення необхідного програмного забезпечення, вибір конкретних бібліотек і сервісів.

До основних плюсів автоматизації можна віднести можливість запускати автотести в будь-який час доби на різних пристроях, що дозволяє проводити автоматичне тестування паралельно з ручним.

Отже, на основі проведеного дослідження було виявлено, що автоматизоване тестування програмного забезпечення за допомогою необхідних інструментів вимагає меншого часу виконання автотестів. Але по вартості відносно більше автоматизоване, аніж ручне. Кожен може зробити висновок для себе, вибираючи різні критерії: час, ціна, чи надійність, проте не забувайте про головний принцип тестування: «Вичерпне тестування неможливе».

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Automation Testing Tutorial: What is Automated Testing. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.guru99.com/automation-testing.html>
2. Винниченко И. В. Автоматизация процессов тестирования / Винниченко И. В. – СПб. : Питер, 2011. – 203 с.
3. Криспин Л. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд: пер. з англ. / Л. Криспин. Д. Грегори – М.: «Вильямс», 2011. – 463 с. – ISBN 0-471-46912-2.
4. Куликов Святослав Тестирование программного обеспечения. Базовый курс/ Святослав Куликов., 2021. – Т. 3 : – 298 с.
5. Burns D. Selenium 2 Testing Tools: Beginner's Guide / Burns D. – Birmingham: Packt Publishing, 2012. – 437 с.
6. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу:  
[https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0\\_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)
7. Обзор протокола HTTP [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/ru/docs/Web/HTTP/Overview>
8. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу:  
[https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F\\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE\\_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)

9. Про Тестинг - Тестирование Программного Обеспечения [Электронный ресурс] – Режим доступа до ресурсу: <http://www.protesting.ru/>
10. An all-in-one test automation solution [Электронный ресурс] – Режим доступа до ресурсу: <https://www.katalon.com/>
11. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects [Электронный ресурс] – Режим доступа до ресурсу:  
<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>
12. Driver requirements Maven [Электронный ресурс] – Режим доступа до ресурсу:  
[https://www.selenium.dev/documentation/en/webdriver/driver\\_requirements/](https://www.selenium.dev/documentation/en/webdriver/driver_requirements/)
13. Getting Started with Selenium for Automated Website Testing [Электронный ресурс] – Режим доступа до ресурсу:  
<https://wiki.saucelabs.com/display/DOCS/Getting+Started+with+Selenium+for+Automated+Website+Testing>
14. Installing Selenium libraries [Электронный ресурс] – Режим доступа до ресурсу:  
[https://www.selenium.dev/documentation/en/selenium\\_installation/installing\\_selenium\\_libraries/](https://www.selenium.dev/documentation/en/selenium_installation/installing_selenium_libraries/)
15. IntelliJ IDEA [Электронный ресурс] – Режим доступа до ресурсу:  
<https://www.jetbrains.com/ru-ru/idea/>
16. Oracle Java [Электронный ресурс] – Режим доступа до ресурсу:  
<https://www.oracle.com/java/>
17. Selenium Webdriver [Электронный ресурс] – Режим доступа до ресурсу:  
[https://drive.google.com/file/d/0B7TBmsv\\_w76nQ1FfQVdyWWRXYk0/view](https://drive.google.com/file/d/0B7TBmsv_w76nQ1FfQVdyWWRXYk0/view)



18. Selenium with Java : Getting Started to Run Automated Tests [Электронный ресурс] – Режим доступа до ресурсу: <https://www.browserstack.com/guide/selenium-with-java-for-automated-test>
19. Strategies for Testing Web Applications from the Client Side [Электронный ресурс] – Режим доступа до ресурсу: <https://cs.fit.edu/media/TechnicalReports/cs-2003-11.pdf>
20. Top 10 Automation Testing Tools in 2021 [Электронный ресурс] – Режим доступа до ресурсу: <https://www.netsolutions.com/insights/top-10-automation-testing-tools/>
21. Welcome to Apache Maven [Электронный ресурс] – Режим доступа до ресурсу: <https://maven.apache.org/>