МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____(*Савченко А.С.*)
«_____»_____2021 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)
ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: *„Автоматизована система тестування інтернет-ресурсів на базі платформи Selenium"*
Виконавець: *Шевцова Юлія Михайлівна*

Керівник: **к.т.н., доцент** *Куклінський Максим Володимирович*

Нормоконтролер: **ст. викл.** *Шевченко О. П.*

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Освітній ступінь: Бакалавр

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____(*Савченко А. С.)*

«____» _____ 2021 р.

ЗАВДАННЯ

на виконання дипломного проекту студентки

Шевцової Юлії Михайлівни

1. Тема дипломного проекту: «Автоматизована система тестування інтернет-ресурсів на базі платформи Selenium» затверджена наказом ректора від «22» квітня 2021 р. № 636/ст.
2. Термін виконання роботи: з 10.05.2021 до 12.06.2021.
3. Вихідні дані до роботи: IntelliJ IDEA, Selenium IDE, Java, JUnit.
4. Зміст пояснювальної записки: характристика видів інтернет ресурсів, види веб-сайтів, характеристика видів, аналіз способів тестування, аналіз існуючих засобів для автоматизованого тестування, переваги та недоліки засобів дщля автоматизованого тестування, розробка та дослідження програмного модуля тестування комерційного веб-сайтую.

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1.	Дослідження проблеми та вивчення електронних джерел	10.05.2021	
2.	Аналіз та дослідження процесу роботи інтернет-ресурсів	12.05.2021	
3.	Аналіз існуючих засобів автоматизаії тестування	18.05.2021	
4.	Розробка програми автоматизації тестування	20.05.2021	
5.	Оформлення пояснювальної записки дипломної роботи	01.06.2021	
6.	Оформлення графічної частини дипломної роботи	02.06.2021	
7.	Підготовка доповіді до захисту дипломної роботи	03.06.2021	
8.	Підготовка до захисту дипломної роботи.	12.06.2021	

Студент (***Шевцова Ю.М.)***

Керівник дипломного проекту (***Куклінський М.В.)***

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Автоматизована система тестування інтерне-ресурсів на базі платформи Selenium» складає 60 с., 22 рис., 2 табл., 12 джерел, 1 додаток.

Об'єкт дослідження: процес тестування програмних продуктів.

Мета роботи: покращення показників якості програмного продукту інтернет-ресурсів шляхом впровадження автоматизованного тестування.

Предмет дослідження: алгоритм автоматизованого тестування інтернет-ресурсу (комерційний веб-сайт) на базі платфоми Selenium.

Методи та дослідження: характеристика та види інтернет-ресурсів, характеристика та аналіз типів веб-сайтів, характеристика видів тестування, способи тестування, аналіз існуючих засобів автоматизації тестування, реалізація автотесту за допомогою платформи Selenium.

Отримані результати та їх новизна: програма автоматизованого тестування, що може використовуватись як самостійний продукт, або бути складовою частиною набору Smoke тестів у системі підтримки тестування програмного забезпечення.

SELENIUM, WEBDRIVER, BDD, PAGE OBJECT, JAVA, ПЛАТФОРМА, ІНТЕРНЕТ-РЕСУРС, WEB-САЙТ, ТЕСТУВАННЯ, АВТОТЕСТ, АВТОМАТИЗАЦІЯ, МЕТОД, ФРЕЙМВОРК, АНОТАЦІЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1. ХАРАКТЕРИСТИКА ІНТЕРНЕТ-РЕСУРСІВ 10
1.1. Поняття інтернет-ресурсу 10
1.2. Види веб-ресурсів 12
1.2.1. Веб-сайти бізнесу 12
1.2.2. Веб-сайти брошур та каталогів 13
1.2.3. Веб-сайти електронної комерції 14
1.2.4. Некомерційні веб-сайти 14
1.2.5. Освітні веб-сайти 15
1.2.6. Веб-сайти бізнес-довідників 15
1.2.7. Веб-сайти порталів 16
1.2.8. Пошукові системи 17
1.2.9. Веб-сайти з краудфандингу 17
1.2.10. Веб-сайти портфоліо/веб-сайти з резюме 18
1.2.11. Інформаційні веб-сайти 18
1.2.12. Веб-сайти соціальних мереж 19
1.2.13. Вікі-сайти 19
1.2.14. Веб-сайти новин та веб-сайти журналів 20
1.2.15. Розважальні веб-сайти 21
1.2.16. Персональні/автобіографічні веб-сайти 22
1.2.17. Веб-сайти блогу 22
Висновки до розділу 1 23
РОЗДІЛ 2. ОСОБЛИВОСТІ ТЕСТУВАННЯ ІНТЕРНЕТ-РЕСУРСІВ 24
2.1. Процес розробки веб-сайтів 24
2.2. Види тестування програмного продукту 27
2.2.1. Функціональні види тестування 28

2.2.2. Нефункціональні види тестування 30
2.2.3. Тестування, пов’язане зі змінами 32
2.3. Способи тестування програмного продукту 33
2.4. Інструментарії для тестування програмного продукту 35
2.5. Мова програмування Java 40
2.6. Середовище для автоматризованого тестування інтернет ресурсів 41
Висновки до розділу 2 41
РОЗДІЛ 3. РЕАЛІЗАЦІЯ АВТОТЕСТУ ЗА ДОПОМОГОЮ SELENIUM ДЛЯ INTERNET-МАГАЗИНУ «CANADIAN TIRE» 43
3.1. Створення тестового сценарію 43
3.2. Запуск автотесту для Internet-магазину «Canadian Tire» 51
Висновки до розділу 3 55
ВИСНОВКИ 56
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 57
ДОДАТКИ 59

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

1. CMS – система керування вмістом (Content Management system).
2. ПЗ – програмне забезпечення.
3. URL – уніфікований локатор ресурсів (Uniform Resource Locator).
4. DSP – платформа попиту (demand side platform).
5. SSP – платформа сторони продажу (Sell Side Platform).
6. BDD – керована поведінкою розробка (Behavior-driven development).
7. HTML – мова гіпертекстової розмітки (HyperText Markup Language).
8. PDF –формат портативного документу (Portble Document Format).
9. WWW – всесвітня мережа (World Wide Web).
10. AOL - американська медакопанія (America Online).
11. MSN – великий інтернет провайдер і веб-портал (Microsoft Network).
12. API – прикладний пргограмний інтерфейс (Application Programming Interface).
13. UFT – уніфіковане функціональне тестування (Unified Functional Testing).
14. CI – неперервна інтеграція (Continuous Integration).
15. IDE – інтегроване середовище розробки (Integrated development environment).
16. HTTP – протокол передачі даних (HyperText Transfer Protocol).
17. WORA – гасло, створене Sun Microsystem для ілюстрації кросплатформених переваг мови Java (Write once, run anywhere - «Напиши один раз – запускай де завгодно»).
18. JVM – віртуальна машина Java (Java Virtual Machine).
19. POM – проектна модель об’є’кта (Project Object Model).
20. PFM – патерн (Page Factory Model).

ВСТУП

Актуальність. Інтернет став невід'ємною частиною нашого життя. Кожен користувач інтернетом може здійснювати покупки, шукати інформацію, переглядати новини, дивитися відео, грати в онлайн-ігри і т.ін. Всі ці дії хдійснюються за допомогою інтернет-ресурсів. Але так, як конкуренція дуже велика, то варто розуміти, що користувачам дуже важливо як функціонал так і зовнішня картинка сайту. Для цього і потрібно виконувати тестування веб-ресурсів, так як помилки функціоналу, незручність користуванням веб-сайтом можуть привести до втрати аудиторії, або втрати даних, якщо помилки будуть у безпеці сайту.

Кожного дня на веб-сайтах занаходять помилки, які приносять власнику величезні фінансові втрати. Для того, щоб уникнути цих моментів, потрібно створювати систему тестування, це зможе зберегти фінанси та, звичайно, репутацію компанії замовника.

Метою дипломного проекту є створення автоматизованої системи тестування на базі платформи Selenium для тестування основного функціоналу інтернет -ресурсу.

Об'єкт дослідження: автоматизована система тестування основного функціоналу інтернет-ресурсу.

Предметом дослідження являється модель програмного рішення для автоматизованого тестування основного функціоналу інтернет-ресурсу.

Новизна даної моделі програмного ріщення полягає в розробці нового комбінованого програмного застосунку, який вирішить питання щодо поєднання засобів та інструментів для автоматизованого тестування основного функціоналу інтернет-ресурсів. Хоча існує безліч інструментаріїв для тестування інтернет-ресурсів, за допомогою яких проводиться тестування, але, на жаль, не існує цілісного програмного рішення для комбінованого тестування основного функціоналу.

У першому розділі було розглянуто види інтернет-ресурсів, які вони бувають, і які найвідоміщі для користувачів. Також було оглянуто та проведено аналіз стовсовно типів веб-сайтів, які користувач може зустріти в інтернеті.

У другому розділі було розглянуто процес розробки веб-сайтів, види тестування, та проаналізовано способи тестування. Також для реалізації було обрано платформу Selenium, шляхом порівняння з іншими засобами для тестування, мову Java, та середовище розробки IntelliJ IDEA.

У третьому розділі було описано створення автоматизованих тестів та прикріплено скріншоти опису всіх класів, а також описана інструкція для запуску тетсів, для вирішення проблем, які можуть виникнути при даній дії і виконана реалізація самих тестів.

РОЗДІЛ 1
ХАРАКТЕРИСТИКА ІНТЕРНЕТ-РЕСУРСІВ

1.1. Поняття інтернет-ресурсу

Всесвітня павутина (World Wide Web) містить численну кількість інтернет-ресурсів. До цих ресурсів відносяться усі доменні імена, електронні адреси, уніфіковані локатори ресурсів (URL), веб-сайти, мобільні програми, бази даних, Інтернет-блоги, сайти соціальних медіа (наприклад, Instagram, YouTube, Twitter), рекламні платформи, DSP(платформа сторони, яка представляє попит на цільову аудиторію, тобто рекламодавця), SSP (платформа сторони, яка надає місце для розміщення реклами), посередники даних (data brokers), та інші онлайн ресурси. Розглянемо такі інтернет-ресурси як веб-концепції: веб-сторінки, веб-сайти, веб-сервер:

- ✓ **Веб-сторінка**

Веб-сторінка - це простий документ, який відображається браузером. Такі документи написані мовою HTML. Браузери можуть також відображати інші документи, такі як файли PDF або зображення, але термін веб-сторінка конкретно стосується документів HTML. В іншому випадку використовується лише термін документ. Веб-сторінка може вбудовувати безліч різних типів ресурсів, таких як:

- інформація про стиль - контроль зовнішнього вигляду сторінки;
- сценарії - які додають інтерактивності сторінці;
- медіа - зображення, звуки та відео.

Усі веб-сторінки, доступні в Інтернеті, доступні за унікальною адресою. Щоб отримати доступ до сторінки, просто введіть її адресу в адресному рядку браузера.

- ✓ **Веб-сайт**

Веб-сайт - це сукупність пов’язаних веб-сторінок (а також пов’язаних з ними ресурсів), які мають унікальне доменне ім’я. Кожна веб-сторінка даного веб-сайту містить чіткі посилання - більшу частину часу у формі частини тексту, на яку можна натиснути, - що дозволяє користувачеві переходити з однієї сторінки веб-сайту на іншу. Щоб отримати доступ до веб-сайту, потрібно його доменне ім’я ввесим в адресний рядок веб-переглядача, і браузер відобразить головну веб-сторінку веб-сайту або домашню сторінку.

Веб-сторінки та веб-сайт, який містить лише одну веб-сторінку особливо легко сплутати. Такий веб-сайт іноді називають односторінковим.

- ✓ **Веб-сервер**

Веб-сервер - це комп'ютер, що розміщує один або кілька веб-сайтів. "Хостинг" означає, що всі веб-сторінки та файли, що їх підтримують, доступні на цьому комп'ютері. Веб-сервер надсилатиме будь-яку веб-сторінку з веб-сайту, який він розміщує, у будь-який браузер за запитом користувача.

Більшість людей плутає веб-сайти та веб-сервери. Наприклад, хтось каже: "Мій веб-сайт не відповідає", це насправді означає, що веб-сервер не відповідає, і тому веб-сайт недоступний. Що ще важливіше, оскільки веб-сервер може розміщувати кілька веб-сайтів, термін веб-сервер ніколи не використовується для позначення веб-сайту, оскільки це може спричинити велику плутанину.Існує багато різних типів веб-сайтів, заснованих на їх цілі та типі організації, для якої вони створені. Розглянемо найпоширеніши типи, з якими стикаються користувачі інтернету [1].

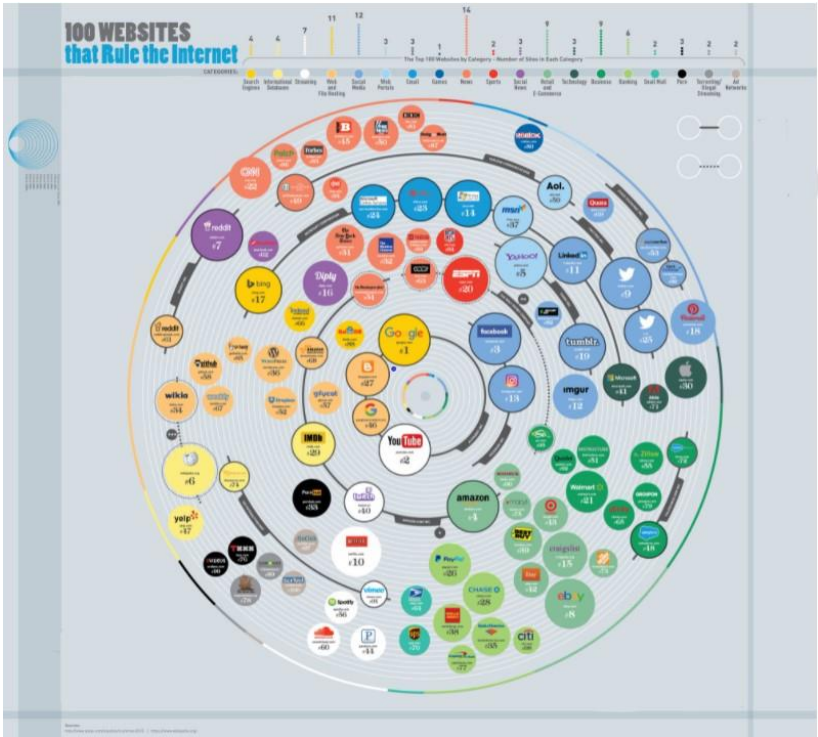


Рис.1.1. Поширені веб-сайти Інтернету

1.2. Види веб-ресурсів

Існує багато різних типів веб-сайтів, заснованих на їх цілі та типі організації, для якої вони створені. На рис.1 зображені найбільш відвідувані сайти інтернету, або як написано на зображені «Сайти, що керують інтернетом». Кожен з сайтів має свій тип, заснований на їх цілі та типі організації, для якої вони створені. Розглянемо найпоширеніші типи, з якими стикаються користувачі інтернету.

1.2.1. Веб-сайти бізнесу

Бізнес-веб-сайт (Business websites) - це веб-сайт, призначений для представлення особистого бізнесу в Інтернеті. Все ще буває так, що не всі активні компанії присутні в Інтернеті. Деякі підприємці можуть отримати достатньо бізнесу завдяки рекомендаціям з вуст в уста і за допомогою контрактів із встановленими контактами, щоб не відчувати потреби рекламувати свої послуги в Інтернеті на

власному веб-сайті. Подібним чином, деякі професіонали можуть шукати роботу на сторонніх веб-сайтах, таких як LinkedIn.

Важливіше мати власний веб-сайт, якщо бізнес має або прагне наймати співробітників, прагне залучати контракти та продажі, рекламуючи себе в Інтернеті, або повинен представляти свої послуги професійно представленим та сучасним способом на цільовому ринку. Все ще часто трапляється, що хтось перевірить ваш веб-сайт, навіть якщо він отримав особисту рекомендацію користуватися вашими послугами. Великі міжнародні корпорації, які не продають безпосередньо населенню, як правило, мають великі глянцеві веб-сайти, призначені для представлення та підтримки свого бренду та корпоративного іміджу.

1.2.2. Веб-сайти брошур та каталогів

Веб-сайт брошури - це веб-сайт, призначений для відображення продуктів та послуг компанії для перегляду в Інтернеті. Як правило, вони включатимуть відгуки, тематичні дослідження, відповідні зображення та відео. Веб-сайт брошури простіший, ніж веб-сайт каталогу, оскільки йому не потрібно перераховувати велику кількість різних товарів, а просто служить привабливо представленою онлайн-рекламою для бізнесу, показуючи, де він знаходиться, як встановити контакт та загальні умови, які товари чи послуги він пропонує. Веб-сайт брошури може бути досить простим і складатися з однієї сторінки (хоча, як правило, не менше п'яти сторінок), тоді як веб-сайт каталогу, як правило, має десятки або сотні сторінок, що представляють різні асортименти товарів та окремі товари.

Веб-сайт каталогу відрізняється від веб-сайту електронної комерції тим, що він не дозволяє здійснювати онлайн-оплату та оплату товарів або послуг. Він призначений для їх демонстрації та спонукає тих, хто переглядає, зв’язатись із бізнесом, щоб зробити замовлення або запросити пропозицію, залежно від того, що пропонується.

1.2.3. Веб-сайти електронної комерції

Веб-сайт електронної комерції - це веб-сайт, розроблений і створений, щоб дозволити клієнтам здійснювати покупки всіх продуктів або послуг компанії безпосередньо в Інтернеті, включаючи здійснення платежів через мережевий шлюз. Це можна зробити в будь-який час дня і ночі, не вимагаючи телефонного дзвінка або іншої форми безпосереднього контакту з бізнесом.

Веб-сайти електронної комерції відображають усі товари або послуги, які можна придбати, а потім дозволяють тим, хто їх переглядає, додати стільки, скільки потрібно, до віртуальних торгових візків, відомих як „кошик". Після того, як покупець додає до кошика все, що він хоче придбати, забезпечується віртуальний процес оформлення замовлення, що дозволяє обробити покупку. Це, як правило, інтегровано або з процесором оплати за допомогою кредитних / дебетових карток, таким як Worldpay, або з віртуальною системою онлайн-платежів, такою як Paypal, яку можна налаштувати для залучення грошей безпосередньо з поточного рахунку або з віртуального балансу. До відомих ринків електронної комерції належать eBay, Amazon Marketplace та Abebooks.

1.2.4. Некомерційні веб-сайти

Некомерційним веб-сайтом є веб-сайт, який представляє некомерційну організацію, таку як благодійна організація в галузі медичних досліджень, зарубіжна благодійна організація або благодійна організація, що займається спадщиною. Хоча багато інших типів веб-сайтів технічно не приносять прибутку, включаючи освітні веб-сайти, що представляють державні та деякі приватні школи, термін некомерційні веб-сайти, як правило, використовують щодо благодійних організацій, крім шкіл та коледжів.

Некомерційні веб-сайти зазвичай представляють роботу організації з точки зору її переваг як для суспільства, так і для осіб, яким надана допомога чи фінансування. Вони, швидше за все, включатимуть подробиці про те, як робити

пожертви та заповіти, а також про те, як подавати заявку на фінансування (якщо це можливо).

1.2.5. Освітні веб-сайти

Навчальний веб-сайт у найвужчому визначенні - це веб-сайт, що представляє навчальний заклад, такий як традиційна школа, коледж чи університет. Він також може представляти приватного постачальника освіти, такого як репетитор, або віртуальний коледж, що пропонує он-лайн та дистанційні курси.

Освітні веб-сайти повинні бути розроблені таким чином, щоб якомога професійніше та привітніше представляти своїх постачальників освіти, щоб вони залучали студентів та запевняли як учнів, так і батьків щодо якості освіти.

Вони не обов'язково містять освітні джерела на самому веб-сайті, але, швидше за все, містять деталі кожного навчального відділу в установі, фотографії містечка (якщо це можливо), дати термінів, збори, адреси директора чи керівника, контактні дані, та новини. Прикладом такого освітнього веб-сайт є веб-сайт Національного авіаційного університету.

1.2.6. Веб-сайти бізнес-довідників

Веб-сайт бізнес-довідників - це веб-сайт, який збирає в одному місці дані про безліч різних підприємств. Такі веб-сайти традиційно були корисними місцями для розміщення реклами, а також слугували цінним джерелом безкоштовних посилань на ділові веб-сайти, хоча все частіше такі посилання, будучи безкоштовними для отримання, знижувались у пошукових системах, таких як Google, коли він оцінює владу домену на основі вхідних посилань.

Деякі довідники підприємств є національними та охоплюють усі сфери бізнесу; інші - місцеві, але все ще охоплюють усі поля; в той час як інші є специфічними для певних сфер бізнесу, і вони іноді є національними, але іноді й

міжнародними. Yell - один з найвідоміших бізнес-довідників Великобританії завдяки своєму походженню в каталозі друку Yellow Pages.

Деякі бізнес-каталоги монетизуються повністю, дозволяючи рекламу третіх сторін, тоді як інші пропонують розширені списки та інші рекламні послуги в обмін на платну передплату.

1.2.7. Веб-сайти порталів

Веб-сайт порталу - це веб-сайт, який спрямовує на велику кількість різнорідних підсайтів або зовнішніх сайтів. Він розроблений, як випливає з назви, як портал або точка входу до іншого вмісту, і, як правило, він намагатиметься залучити великий обсяг трафіку, посилаючись на широкий спектр цікавого вмісту та інших функцій. Такі веб-сайти можуть також виконувати роль вибраних агрегаторів вмісту, оскільки вони збираються разом і відображають у формі попереднього перегляду матеріал, який більш повно відображається на підсайтах або зовнішніх сайтах, на які посилаються попередні перегляди.

Найуспішніші веб-сайти порталів історично включали цілу функціональність пошукової системи WWW, будь то на своїх домашніх сторінках або на спеціальному підсайті. Включення, як правило, було основною рушійною силою великого обсягу трафіку і суттєво сприяло тому, що вони мали дуже високий рейтинг сторінки або домен.

Найвідоміші веб-сайти порталів із інтегрованими пошуковими системами включають Yahoo, AOL та MSN. Але є багато інших типів портальних веб-сайтів без пошукових систем, які обслуговують більш нішеві сфери інтересів. Серед функцій, на які посилаються веб-сайти порталу, можуть бути онлайн-ігри, новини, онлайн-книги, веб-сайти спільнот та форумів, музика, відео та статті, що розміщуються зовні на певні теми.

1.2.8. Пошукові системи

Пошукові системи - це ресурси, що служать ключами для пошуку відповідного вмісту в Інтернеті. Ці платформи відрізняються від порталів тим, що вони не поширюють куратори або відібрані керівництвом ресурси кожному, хто їх відвідує. Натомість вони починають з подання чистого аркуша у вигляді вікна пошукового запиту, в яке користувач повинен набрати текст або (у випадку деяких пошукових систем зображень) вставити малюнок. Якщо йдеться про введення тексту, пошукова система запрограмована відповісти на слово користувача, фразу, ряд слів або речення, а потім інтерпретувати намір запиту, перш ніж обробити його за допомогою алгоритму, який прагне представити найбільш релевантні веб-сайти або результати зображення.

Коли зображення вставляється, пошукова система запрограмована на пошук того самого зображення та подібних зображень в Інтернеті. Це може бути корисним для цілей ідентифікації художника чи фотографа, відповідального за невизнане зображення, або для пошуку подібних зображень для використання в конкретному проекті.

Окрім пошукових систем загального призначення, існують також інші зі специфічними цілями, такі як ті, що призначені для пошуку певних видів товарів для продажу. Пошукові системи для книг - один із найвідоміших прикладів, серед яких Bookfinder, Addall, Vialibri та Eurobuch є одними з найбільших гравців на ринку. Видимість пошукової системи - це дуже важлива мета для будь-якого бізнесу, який прагне залучити нових клієнтів в Інтернеті, і саме тут значення розробки пошукової системи відіграє важливу роль у розробці бізнес-сайту.

1.2.9. Веб-сайти з краудфандингу

Веб-сайти з краудфандингом (Crowdfunding websites) - відносно недавнє явище, яке стало важливою частиною веб-економіки. Вони варіюються від веб-сайтів, розроблених для того, щоб люди могли звертатися з особистими

благодійними закликами до своїх друзів та сімей, таких як медичне обслуговування, до веб-сайтів, розроблених для того, щоб дозволити малому бізнесу та новим компаніям розробляти новий продукт та виводити його на ринок. Між ними є такі, як Patreon, які дозволяють людям обіцяти постійну фінансову підтримку досліднику, постачальнику контенту або іншій особі.

1.2.10. Веб-сайти портфоліо/веб-сайти з резюме

Портфоліо-сайти - це веб-сайти, що служать демонстрацією робіт художників, письменників, ремісників, дизайнерів та інших, що працюють у творчих сферах. Вони включатимуть галерею зразків роботи людини, якщо робота носить візуальний характер. Якщо твір має літературний характер або є іншим чином у письмовій формі, веб-сайти портфоліо включатимуть витяги або зразки письмової роботи особи, а також інформацію про будь-які книги та інші публікації, які видав письменник.

Веб-сайт з резюме - це простіший веб-сайт, який представляє професійне життя особи шляхом ілюстрування її історії праці та кваліфікації. Не всі працюють у творчих сферах, що дозволяють відображати їх роботи в портфоліо.

Обидві форми веб-сайту можуть використовуватися для підтримки заявок на роботу та для того, щоб допомогти людині залучити іншу роботу. Оскільки їх основна увага приділяється професіоналам, інформація про особисте життя людини, ймовірно, буде обмеженою, і тон, як правило, буде відносно формальним, хоча деякі згадки про певні інтереси можуть бути зроблені настільки, що це може сприйматися як прояв особистості спосіб, який може допомогти залучити роботодавців.

1.2.11. Інформаційні веб-сайти

Веб-сайти, які присвячені наданню інформації на певну тему чи діапазон тем, але не продають безпосередньо або не пропонують інші курси, називаються інформаційними веб-сайтами. Інформаційні веб-сайти часто представляють хобі чи

інтерес творця, але найпопулярніші можуть залучати фінансування за рахунок доходів від реклами, обіцянок постійної підтримки на таких сайтах, як Patreon, та добровільних одноразових пожертв. Деякі інформаційні веб-сайти на такі теми, як політика, можуть містити сильні думки, які викликають коментарі та суперечки, тоді як інші можуть бути науковими та орієнтованими на історію, літературу чи науку, а треті - практичні. Інформаційні веб-сайти також включають сайти фанатів, присвячені певним громадським діячам та знаменитостям, таким як спортсмени, музиканти, письменники та актори.

1.2.12. Веб-сайти соціальних мереж

Веб-сайт у соціальних мережах призначений для сприяння соціальній взаємодії між людьми, що користуються Інтернетом. Це може мати декілька форм, включаючи публічні чати, особисті послуги обміну миттєвими повідомленнями та знайомі персоналізовані стрічки новин із коментарями та оновленнями новин, розміщені контактами, з якими було укладено взаємну угоду, або приватними особами та компаніями, яких одностороння обрали 'слідувати'. Чати та послуги обміну миттєвими повідомленнями були набагато більш поширеними в 1990-х - на початку 2000-х. Стиль подачі новин для соціальних мереж, обмежений персональними зв’язками, не став помітним до появи Myspace у 2003 році, пізніше пішли Facebook, Twitter та LinkedIn, а ще нещодавно Instagram і Snapchat. Деякі веб-сайти соціальних медіа (наприклад, Facebook) також сьогодні мають функціональність форуму.

1.2.13. Вікі-сайти

Веб-сайт Wiki - це веб-сайт, який створено для того, щоб служити енциклопедичною презентацією інформації, яку можуть внести та відредагувати численні добровольці. Загальний термін походить від його первісного використання найбільшою з усіх, Вікіпедією.

Хоча в принципі редагування відкрите для всіх, кожен веб-сайт Wiki, як правило, має свій власний набір правил, що накладаються менеджерами та застосовуються адміністраторами. Це необхідно для того, щоб підтримувати стандарти якості та допускати арбітраж, заснований на узгодженому наборі принципів, у разі суперечки щодо редагування між різними добровольцями.

Кожна Вікі може також мати свої правила щодо того, який вміст для неї підходить. Навіть сама енциклопедична Вікіпедія за дизайном, Вікіпедія, виключає за політикою будь-яких осіб, які не відповідають її критеріям „помітності" та деяких тем, які не відповідають її критеріям наукової та фактичної надійності.

Інші Вікі можуть бути обмежені лише нішовою темою, щоб глибше зосередитись на ній і зменшити ризик пошкодження та руйнівних редагувань від тих, хто не знає про неї, як можна було б очікувати від загальної енциклопедії.

Загальним правилом усіх веб-сайтів Wiki є те, що вони повинні працювати на некомерційній основі та без реклами. Однак вони можуть приймати пожертви, і там, де їх достатньо, керівникам можуть виплачувати зарплату. Це можливо, можливо, лише на найбільш гучних Вікі. Більшість інших - це не оплачувана праця любові до всіх учасників, а не лише до волонтерів.

1.2.14. Веб-сайти новин та веб-сайти журналів

Через майже миттєве оновлення та доступність Інтернету, він надзвичайно добре піддається випуску новин. Історії можна публікувати та оновлювати в режимі реального часу, охоплюючи аудиторію швидше, ніж заплановані телевізійні чи радіовипуски, і набагато швидше, ніж друковані газети. Насправді веб-сайти новин стали настільки відвідуваними, що вся індустрія друкованих газет за останні 20 років впала у тривалий і стрімкий спад.

Друковані журнали також постраждали в значній мірі від розповсюдження вмісту в Інтернеті, причому навіть ті, хто заорює вузькоспеціалізовані нішеві борозни, як правило, спостерігали зниження своїх тиражів у порівнянні з двадцятьма роками раніше. Однак більшість із них підтримують веб-сайти для

відображення обмеженого діапазону їх вмісту, і це повинно допомогти залучити нові друковані підписки, а також дешеві електронні підписки, які стають дедалі популярнішою альтернативою.

1.2.15. Розважальні веб-сайти

Розважальні веб-сайти - це ті, які призначені для розваги користувача веб-сайту. Вони включають веб-сайти, засновані на комедії, а також веб-сайти, що містять комікси, художні твори, музику, фільми та драматичні серіали. Вони також включають такі, що дозволяють користувачам грати в онлайн-ігри.

Хоча аудіо-візуальні функції зазвичай можна знайти на розважальних веб-сайтах, вони не обов'язково потрібні. Веб-сайти, що дозволяють користувачам читати романи, захищені авторським правом, привабливо представлені у спеціальних програмах для читання, все ще виконують функцію розваги.

Бізнес-моделі розважальних веб-сайтів дуже різняться. Деякі можуть запросити користувачів завантажувати власний вміст, тоді як інші укладають домовленості з видавцями про їх вміст. У випадках веб-сайтів, які дозволяють користувачам завантажувати власний вміст, виникали певні проблеми з піратством та несанкціонованим завантаженням захищеного авторським правом вмісту, що залишалося джерелом широкомасштабних суперечок.

Телевізійні компанії, кінокомпанії та звукозаписні компанії, як правило, мають власні веб-сайти для бізнесу, деякі з яких містять зразок розважального контенту або внутрішні магазини, які його продають.

Netflix - приклад веб-сайту для трансляції фільмів, який став міжнародним гігантом у цій галузі, після того як спочатку злетів головним чином у Каліфорнії.

Spotify сьогодні відомий у всьому світі як веб-сайт з потоковою передачею музики і працює на основі стягування платної щомісячної передплати з користувачів. Потім платять дуже малу комісію звукозаписуючим компаніям за кожну передачу композиції виконавця на платформу.

YouTube - це найвідоміший веб-сайт для потокового відео, який дозволяє власникам авторських прав на комерційні музичні відео та кадри відеозаписів розміщувати їх для безкоштовного завантаження користувачами в обмін на невелику частку доходу від реклами. Це також дозволяє звичайним користувачам завантажувати власні матеріали. Численні заяви про порушення авторських прав на завантажені користувачем матеріали призвели до прийняття автоматизованих перевірок деяких типів вмісту, захищеного авторським правом.

1.2.16. Персональні/автобіографічні веб-сайти

Персональний веб-сайт - це будь-який веб-сайт, який представляє особу особи та/або сім'ї цієї особи. Він може містити елементи блогу, але це більше, ніж просто блог. Він буде містити статичні сторінки з автобіографічними замальовками та спогадами, а також, ймовірно, фотоальбоми та, можливо, щоденникові записи чи інші документи.

На відміну від веб-сайту з портфоліо або резюме (описані вище) особистий веб-сайт не спрямований переважно або виключно на представлення професійної особистості особи або пошуку роботи, хоча він може включати елементи змісту про кар’єру особи.

1.2.17. Веб-сайти блогу

Веб-сайт блогу (Blog websites) - це будь-який веб-сайт, який складається виключно із серії хронологічно впорядкованих публікацій окремою особою. Вони будуть важкими для тексту, але можуть містити і зображення. Хоча їх, швидше за все, можна знайти на особистих субдоменах популярних веб-сайтів, присвячених ведення блогу, таких як Wordpress, Blogspot та Blogger, вони також можуть існувати в незалежних доменах, хоча в цьому випадку блоги частіше є властивістю особистого веб-сайт, а не єдиний фокус.

Найбільш успішні блоги залучають коментарі та публікації статей і можуть з часом створити значний авторитет домену. Після встановлення високого рівня відвідуваності та доброї волі багато власників блогів бачать можливість монетизувати свої блоги, рекламуючи через них свої послуги. Прикладом такого сайта є Going Zero Waste – це екологічний блог, створений Кетрін Келлогг, прихильницею екологічно чистого способу життя без використання пластику. Вона веде блог про нульові відходи [2].

Висновки до розділу 1

В даному розділі було розглянуто інтернет-ресурси, які з них є найбільш відомими користувачам Інтернету, а також було проаналізовано всі види веб-ресурсів та обрано для розгляду тестування комерційний сайт, так як одним з найчастіше, після розважальних сайтів, відвідуваним сайтом є інтернет-магазини, які повинні функціонувати правильно, для цього і важливо тестувати такі веб-ресурси, аби не виникало проблем у користувачів з замовленням, переглядом, регістрацією/авторизацією на сайті та, особливо, оплатою покупок.

РОЗДІЛ 2
ОСОБЛИВОСТІ ТЕСТУВАННЯ ІНТЕРНЕТ-РЕСУРСІВ

2.1. Процес розробки веб-сайтів

Існує сім кроків, яких слід виконувати від початку до кінця при розробці веб-сайту.

1. Дослідження та постановка цілей

Як і в будь-якому проекті, перед початком роботи важливо правильно провести дослідження та встановити цілі. Встановивши цілі, це допоможе веб-сайту отримати напрямок, а також допоможе досягти конкретних досягнень. Процес планування та постановки цілей може зайняти близько 1-2 тижнів. Є кілька питань, які ви повинні задати собі на цьому етапі: Яка ціль мого веб-сайту? На яку аудиторію хотів би орієнтуватися? Які основні цілі веб-сайту?

2. Планування

Планування веб-сайту передбачає створення каркасу та мапи сайту. Це важливий крок, оскільки він нагадує скелет сайту. Цей процес може зайняти близько 2-6 тижнів. Карта сайту дозволяє розробнику отримати схему того, як буде виглядати сайт, які сторінки будуть і як вони будуть взаємодіяти між собою. Це не тільки допомагає планувати, але й корисно для користувацького досвіду.

Користувач повинен мати можливість легко орієнтуватися на сайті, і це починається з розробки мапи сайту. Перш ніж почати планувати вміст, карта сайту дозволяє вам розробити, як виглядатиме структура. Після того, як мапа сайту буде завершена, іншою частиною цього кроку є створення каркасу або макету. Це лише наочні зображення того, як буде виглядати сайт.

3. Проектування макета

Деталі макета - це те, що надасть веб-сайту характер. Це крок творчості із зображеннями, відео та типом речей, які клієнт помітить, коли зайде на сайт. Цей процес може зайняти близько 4-12 тижнів від початку до кінця. Терміни залежать від досвіду, часу, витраченого на проект, і наскільки ретельний розробник. На цьому етапі особливо важливо постійно посилатися на цільову аудиторію. Розглядяються кольори, логотипи та все, що спонукає аудиторію взаємодіяти з веб-сайтом.

4. Написання вмісту

Цей крок може йти одночасно з іншими етапами планування. Письмовий зміст веб-сайту так важливий для його успіху. Хоча цей крок може відбуватися під час інших кроків, він є ключовим і заслуговує на великий досвід. Це може зайняти від 5 до 15 тижнів. Письмовий вміст на веб-сайті допоможе відвідувачеві визначити подальші кроки. Дуже важливо залучити клієнтів і утримати їх. Працюючи над змістом веб-сайту, слід багато чого врахувати. Веб-сайт повинен мати словниковий запас, який пересічна людина може зрозуміти.

Знову ж таки, врахування цільової аудиторії надзвичайно важливо, особливо якщо мова йде про текст, який використовується для надання інформації клієнтам. Це може визначити, які слова та голос будуть використані при написанні, що може створити настрій змісту, хорошим він чи поганим. Це також передбачає створення привабливих назв та заголовків, щоб залучити людей.

5. Кодування

Кодування, як правило, починається з домашньої сторінки і поступово розгалужується на інші сторінки, включені на сайт. Тут потрібно буде дотримуватися карти сайту, щоб переконатися, що все правильно закодовано. Етап кодування може зайняти від 6 до 15 тижнів, залежно від того, скільки вмісту та хитромудрості хотів би замовник. Також важливо встановити фреймворки та CMS, щоб переконатися, що все вміщується на сервері під час процесу встановлення.

Після того, як веб-сайт викладений відповідно до карти сайту, його слід протестувати, перш ніж рухатись далі. Якщо все працює добре, тоді слід додати

решту вмісту та завершити форматування. Цей етап передбачає глибоке розуміння технології, яка використовується.

6. Тестування та запуск

Перш ніж запустити веб-сайт, важливо, щоб його тестувальники зі сторони користувача. Усі посилання та вміст слід перевірити, чи працює це. Також можна використати бета-тестування, тобто тестування користувачів, яке можна пройти, щоб переконатися, що веб-сайт надає користувачам те, що їм потрібно для успішного тестування.

Знову ж таки, є інструменти, за допомогою яких можна визначити, чи потрібно щось змінювати. Обов'язково потрібно перевірити весь письмовий вміст, включаючи орфографію та граматику. Якщо веб-сайт має форми, переконайтися, що вони також працюють правильно. Це можуть бути важливі способи, за допомогою яких користувачі можуть зв'язатися або підписатись на сповіщення та повідомлення. Якщо вони не працюють належним чином, користувачеві може бути дуже важко, а також замовник може через це втратити аудиторію сайту.

Не просто перевірити веб-сайт один раз, а потрібно перевірити його кілька разів. Коли команда розробки та замовник впевнені, що все працює, то можна буде продовжувати і запускати веб-сайт в реліз, завантаживши його на сервер.

7. Технічне обслуговування

Можна подумати, що робота виконана після запуску веб-сайту, але це не так. Оскільки технології та продукти змінюються швидше, ніж будь-коли раніше, важливо бути в курсі того, що відбувається в Інтернеті. Утримання веб-сайту - це важка робота, але чим більше зусиль докладено до його обслуговування, тим краще. Є кілька різних частин для обслуговування веб-сайту.

З одного боку, його слід постійно перевіряти на наявність помилок. Коли користувач стикається з помилкою, це може викликати розчарування і може змусити їх знайти те, що вони шукають десь ще. Помилки також можуть повністю заблокувати їм інформацію, необхідну для прийняття рішення про придбання товару чи послуги. Ось чому важливо не лише перевірити свій веб-сайт на досвід користувачів перед запуском, а й після [3].

Іншим важливим аспектом обслуговування веб-сайту є забезпечення постійного оновлення всього вмісту. Якщо надавати погану або застарілу інформацію користувачам, то можна їх втратити.

Так як тестування виявляється одним із основних факторів розробки сайту та запуску його в інтернет, то потрібно дослідити всі види тестування та методології, які використовуються для цього процесу.

2.2. Види тестування програмного продукту

На сьогоднішній день не існує загальновизнаного визначення "типу тестування програмного забезпечення". Нерідкі випадки, коли методи, рівні або навіть техніка проектування тестів визначаються як тип тестування. Наприклад, іноді тестування білих ящиків, інтеграційне тестування або навіть граничне тестування розглядається як типи тестування.

Під час виконання тестового випадку тестувальник фокусується на певній тестовій цілі. Залежно від його цілей існує три типи тестування програмного забезпечення:

- Функціональне тестування
- Нефункціональне тестування
- Тестування, пов’язане зі змінами

Звичайно, існує безліч підтипів тестування. Ви можете побачити їх на малюнку нижче:

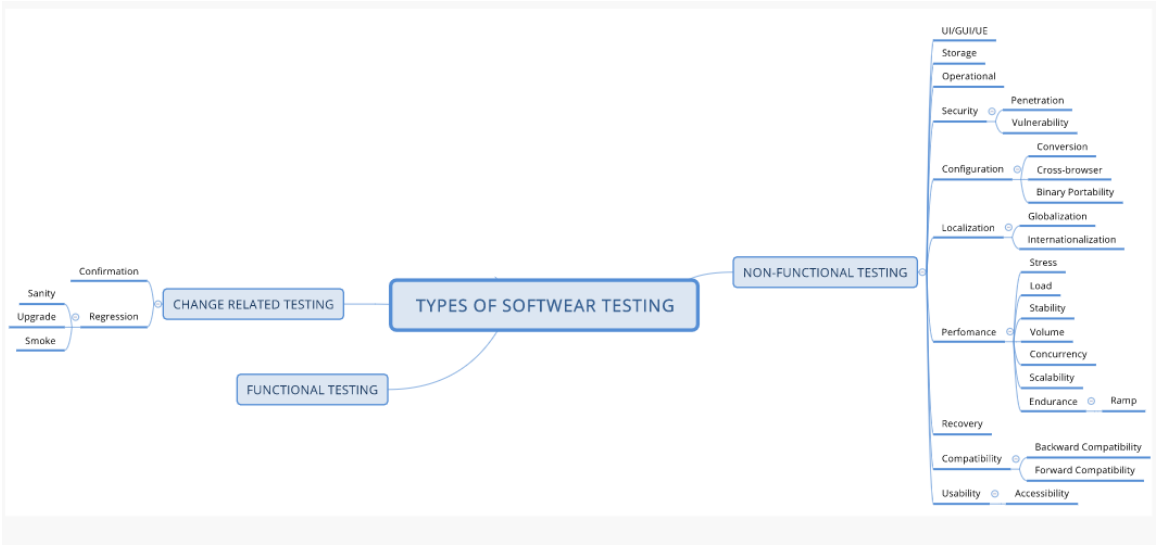


Рис. 2.1. Підтипи видів тестування програмного продукту

2.2.1. Функціональні види тестування

Функціональне тестування підтверджує, що кожна функція програмного додатка працює відповідно до специфікації вимог. Функціональне тестування показує “Що робить система”. Мета цього тестування - перевірити, чи функціонально система досконала.

Існує два ракурси, в яких функціональність тестування можна здійснити за допомогою тестування на основі вимог та тестування на основі бізнес-процесів.

Тестування на основі вимог проводиться у суворій відповідності до визначених вимог.

Тестування на основі бізнес-процесів проводиться відповідно до знань, що базуються на повсякденному комерційному використанні системи.

Переваги функціонального тестування:

1. Функціональне тестування імітує фактичне використання системи.
2. Воно виконується в умовах, близьких до умов замовника.
3. При проведенні функціонального тестування не робиться жодних припущень про структуру системи.
4. Зробити тестування вручну легко.

Обмеження функціонального тестування:

1. Існує велика можливість надмірного тестування.

2. Логічні помилки в програмному забезпеченні можуть бути пропущені при проведенні функціонального тестування.

Найпоширеніші види функціонального тестування:

- Тестування безпеки (Security testing);

Тестування безпеки - це тип тестування програмного забезпечення, який виявляє вразливі місця, загрози та ризики програмних додатків та запобігає зловмисним атакам хакерів. Метою тестування безпеки є виявлення всіх можливих слабких місць програмної системи, які можуть призвести до втрати інформації, доходів, репутації.

- Тестування функцій (Functional testing)

Тестування функцій - це тип тестування програмного забезпечення, який перевіряє програмну систему на відповідність функціональним вимогам/специфікаціям. Метою функціональних тестів є тестування кожної функції програмного забезпечення шляхом надання відповідного введення та перевірки результату на відповідність функціональним вимогам.

Тестування функцій в основному передбачає тестування чорних ящиків, і його не турбує вихідний код програми. Це тестування перевіряє користувальницький інтерфейс, API, базу даних, безпеку, зв'язок між клієнтом та сервером та інші функції тестованої програми. Тестування може проводитися як вручну, так і за допомогою автоматики.

- Тестування взаємодії (Interoperability Testing).

Це функціональне тестування, яке перевіряє здатність додатку взаємодіяти з одним і більше компонентами або системами. Тестування взаєиодії включає в себе тестування сумісності (Compatibility testing) та інтеграційне тестування (Integration testing).

Тестування сумісності (Compatibility testing) - це тип тестування програмного забезпечення, щоб перевірити, чи може ваше програмне забезпечення працювати на різному обладнанні, операційних системах, додатках, мережевому середовищі чи мобільних пристроях [4].

Інтеграційне тестування - це тип тестування, де програмні модулі інтегровані логічно та перевіряються як група. Типовий програмний проект складається з безлічі програмних модулів, кодованих різними програмістами. Метою цього рівня тестування є виявлення дефектів взаємодії між цими програмними модулями при їх інтеграції. Тестування інтеграції зосереджується на перевірці передачі даних між цими модулями.

2.2.2. Нефункціональні види тестування

Типи нефункціонального тестування стосуються нефункціональних вимог. Нефункціональне тестування допомагає оцінити готовність системи відповідно до різних критеріїв, які не охоплюються функціональним тестуванням. На відміну від функціонального тестування, воно показує “Наскільки добре працює система”.

Для більшого розуміння цього виду тестування варто розглянути підтипи.

- Тестування продуктивності (Performance Testing)

Тестування продуктивності - це процес тестування програмного забезпечення, що використовується для тестування швидкості, часу відгуку, стабільності, надійності, масштабованості та використання ресурсів програмного додатку за певного навантаження. Основною метою тестування продуктивності є виявлення та усунення вузьких місць продуктивності програмного додатку. Також тестування продуктивності включає в себе такі типи тестування:

- Навантажувальне тестування (Load Testing);

Тестування навантаження - це нефункціональний процес тестування програмного забезпечення, при якому продуктивність програмного забезпечення перевіряється за певного очікуваного навантаження. Він визначає, як поводиться програмний додаток при одночасному доступі декількох користувачів. Метою тестування навантаження є покращення вузьких місць продуктивності та забезпечення стабільності та безперебійного функціонування програмного забезпечення перед розгортанням.

- Стрес - тестування (Stress Testing);

Стрес-тестування - це тип тестування програмного забезпечення, який перевіряє стабільність та надійність програмного забезпечення. Метою стрес-тестування є вимірювання програмного забезпечення на його надійність та можливості обробки помилок в умовах надзвичайно великих навантажень, а також забезпечення того, щоб програмне забезпечення не аварійно працювало в ситуаціях кризи. Він навіть перевіряє нормальні робочі точки та оцінює, як працює програмне забезпечення в екстремальних умовах.

- Тестування стабільності (Stability Testing);

Тестування стабільності - це тип нефункціонального тестування програмного забезпечення, що проводиться для вимірювання ефективності та здатності програмного додатку безперервно функціонувати протягом тривалого періоду часу. Метою тестування на стабільність є перевірка того, чи програмний додаток у будь-який момент часу виходить з ладу або виходить з ладу при звичайному використанні.

- Тестування установки (Installation testing);

Робота із забезпечення якості, яка фокусується на тому, що потрібно зробити клієнтам, щоб успішно встановити та налаштувати нове програмне забезпечення. Це може включати повне, часткове або оновлення процесів встановлення, і зазвичай це робиться інженером з тестування програмного забезпечення спільно з менеджером конфігурації.

- Тестування зручності користування (Usability Testing);

Тестування зручності користування - це тип тестування, який перевіряє наскільки легко та зручно користуватися програмним продуктом. В основному при цьому тестуванні перевіряється чи зручний інтерфейс, навігація, зрозумілість, чи немає мільйона кроків для виконання простої операції.

- Тестування на відновлення (Recovery Testing);

Тестування відновлення - це техніка тестування програмного забезпечення, яка перевіряє здатність програмного забезпечення відновлювати такі збої, як збої програмного / апаратного забезпечення, відмови мережі тощо. Метою Тестування відновлення є визначення того, чи можна продовжувати операції з програмним

забезпеченням після катастрофи або втрати цілісності. Тестування відновлення передбачає повернення програмного забезпечення до точки, коли була відома цілісність, та повторну обробку транзакцій до точки відмови.

- Тестування конфігурації (Configuration Testing).

Тестування конфігурації - це техніка тестування програмного забезпечення, при якій програмне забезпечення тестується за допомогою декількох комбінацій програмного та апаратного забезпечення, щоб оцінити функціональні вимоги та з'ясувати оптимальні конфігурації, за якими програмний додаток працює без будь-яких дефектів та недоліків [4].

2.2.3. Тестування, пов’язане зі змінами

Проводиться тестування, пов’язане зі змінами, щоб переконатись, що раніше усунуті помилки було виправлено, та виявити помилки, які, можливо, випадково з’явились у новій версії. Відповідно до цих цілей існує 4 підтипи тестування:

- Димове тестування (Smoke testing)

Димове тестування - це технологія тестування програмного забезпечення, що виконується після побудови програмного забезпечення, щоб переконатись, що критичні функції програмного забезпечення працюють нормально. Він виконується перед виконанням будь-яких детальних функціональних або регресійних тестів. Основною метою тестування диму є відмова від програмного забезпечення з дефектами, щоб команда контролю якості не витрачала час на тестування зламаного програмного додатку.

- Regression testing

Регресійне тестування визначається як тип тестування програмного забезпечення для підтвердження того, що нещодавня зміна програми або коду не вплинула негативно на існуючі функції. Це не що інше, як повний або частковий вибір уже виконаних тестових кейсів, які повторно виконуються, щоб забезпечити нормальну роботу існуючих функціональних можливостей.

Це тестування проводиться, щоб переконатися, що нові зміни коду не повинні мати побічних ефектів на існуючі функціональні можливості. Це гарантує, що старий код все ще працює після внесення останніх змін коду.

- Тестування на обґрунтованість (Sanity testing)

Тестування на обґрунтованість, або як ще називають санітарне тестування - це свого роду тестування програмного забезпечення, яке проводиться після отримання збірки програмного забезпечення, з незначними змінами в коді чи функціональних можливостях, щоб переконатися, що помилки виправлені, і подальші проблеми не виникають через ці зміни. Мета полягає в тому, щоб визначити, що запропонована функціональність працює приблизно так, як очікувалося. Якщо перевірка розумності не вдається, збірка відхиляється, щоб заощадити час і витрати, пов'язані з більш суворим тестуванням.

У деяких джерелах помилково вважають, що санітарне та димове тестування - це одне і теж. Ці види тестування мають "вектора руху", напрямки в різні боки. На відміну від димового (Smoke testing), тестування обгрунтованості (Sanity testing) направлено вглиб перевіряється функції, в той час як димове направлено в шир, для покриття тестами якомога більшого функціоналу в найкоротші терміни [5].

2.3. Способи тестування програмного продукту

Існує 3 способи тестування.

1. Тестування вручну (Manual testing) - це найбільш практичний тип тестування, який у певний момент застосовується кожною командою. Звичайно, у сьогоднішньому стрімкому життєвому циклі розробки програмного забезпечення ручне тестування важко масштабувати.

2. Автоматизоване тестування (Automated testing) використовує тестові сценарії та спеціалізовані засоби для автоматизації процесу тестування програмного забезпечення.

3. Напівавтоматизованого тестування (Semi automated testing) – суміі попередніх двох видів (наприклад, за допомогою інструменту автоматизації

створюємо аккаунт користувача в системі, а потім вручне створюємо покупки в інтернет-магазині)

Варто розглянути переваги та недоліки найбільш використовувані з них: тестування вручну та автоматизоване тестування.

Переваги ручного тестування:

1. Тестування вручну можна проводити на всіх видах додатків.
2. Переважно для продуктів з коротким життєвим циклом.
3. Нещодавно розроблені тестові кейси слід виконувати вручну.
4. До автоматизації додаток потрібно протестувати вручну.
5. Краще в проектах, де вимоги часто змінюються, а також у продуктах, де графічний інтерфейс постійно змінюється.
6. З точки зору початкових інвестицій це дешевше порівняно з тестуванням автоматизації.
7. Потрібно менше часу та витрат, щоб розпочати продуктивне ручне тестування.
8. Це дозволяє тестувальнику проводити спеціальне тестування.
9. Немає необхідності у тестувальника мати знання про засоби автоматизації.

Недоліки ручного тестування:

1. Ручне тестування займає багато часу, в основному, під час проведення регресійного тестування.
2. Тестування вручну менш надійне порівняно з тестуванням автоматизації, оскільки воно проводиться людьми. Тож завжди будуть схильні до помилок і помилок.
3. Дороге над автоматичним тестуванням у довгостроковій перспективі.

Переваги автоматизованого тестування:

1. Тестування автоматизації виконується швидше.
2. Це дешевше порівняно з ручним тестуванням у довгостроковій перспективі.
3. Автоматизоване тестування є більш надійним.

4. Автоматизоване тестування є більш потужним і універсальним.
5. Воно в основному використовується для регресійного тестування.
6. Воно може бути багаторазовим, оскільки процес автоматизації може бути записаний.
7. Не вимагає втручання людини. Тестові сценарії можна запускати без нагляду.
8. Це допомагає збільшити охоплення тестом.

Недоліки автоматизованого тестування:

1. Рекомендується лише для стабільних продуктів.
2. Тестування автоматизації спочатку дороге.
3. Більшість засобів автоматизації дорогі.
4. Він має деякі обмеження, такі як обробка капчу, отримання візуальних аспектів інтерфейсу, таких як шрифти, колір, розміри тощо.
5. Величезне обслуговування у разі неодноразових змін вимог [6].

2.4. Інструментарії для тестування програмного продукту

Звичайно тестування відбувається за допомою різноманітних інструментаріїв, які необхідні для вдосконалення програми або програмного забезпечення , тому варто розглянути найвідоміші з них.

1. *Katalon Studio* - це автоматизована платформа для тестування, яка пропонує повний набір функцій для реалізації повних автоматизованих рішень для тестування Інтернету, API та мобільних пристроїв. Побудований поверх фреймворків із відкритим кодом Selenium та Appium, Katalon Studio дозволяє командам швидко розпочати роботу з автоматизацією тестів, зменшуючи зусилля та досвід, необхідні для навчання, та інтегруючи ці фреймворки для потреб автоматизованого тестування.

2. *Selenium* - чи не найпопулярніша система автоматизації, яка складається з безлічі інструментів та плагінів для тестування веб-додатків. Селен відомий своєю потужною здатністю підтримувати тестування продуктивності веб-додатків. Селен є

популярним вибором у просторі автоматизації тестів з відкритим кодом, частково завдяки великій та активній розробці та спільноті користувачів. Однак його крута крива навчання є недоліком, який заважає продуктивності багатьох команд.

3. *Уніфіковане функціональне тестування (UFT)* є, мабуть, найпопулярнішим комерційним інструментом для автоматизації функціональних тестів. UFT пропонує повний набір функцій, які можуть покрити більшість функціональних потреб автоматизованого тестування на настільних, мобільних та веб-платформах.

4. *TestComplete* - це також комерційна інтегрована платформа для тестування настільних, мобільних та веб-додатків. Як і UFT, TestComplete пропонує ряд ключових функцій автоматизації тестування, таких як тестування на основі ключових слів та даних, крос-браузерне тестування, тестування API та інтеграція CI. Цей інструмент підтримує низку мов, включаючи JavaScript, Python, VBScript, JScript, DelphiScript, C ++ Script та C # Script для написання тестових скриптів.

У таблиці нижче наведено порівняння інструментів, заснованих на ключових особливостях автоматизації програмного забезпечення:

Таблиця 2.1

Порівняльна характерисика інструментів

Особливості	**Katalon Studio**	**Selenium**	**UFT**	**TestComplete**
Платформа розробки тестів	Крос-платформеність	Крос-платформеність	Windows	Windows
Мови сценаріїв	Java/Groovy	Java, C#, Perl, Python, JavaScript, Ruby, PHP	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++, and C#

Продовження таблиці 2.1

Простота установки та використання	Простота налаштування та запуску	Потрібно встановити та інтегрувати різні інструменти	Простий в налаштуванні та запуску	Простий в налаштуванні та запуску
Тестування на основі зображень	Вбудована підтримка	Потрібно встановити додаткові бібліотеки	Вбудована підтримка, розпізнавання об'єктів на основі зображень	Вбудована підтримка
Постійні інтеграції	Popular CI tools (e.g. Jenkins, Teamcity)	Various CI tools (e.g. Jenkins, Cruise Control)	Various CI tools (e.g. Jenkins, HP Quality Center)	Various CI tools (e.g. Jenkins, HP Quality Center)
Тип ліцензії	Безкоштовна програма	Відкритий код (Apache 2.0)	Запатентована	Запатентована
Вартість	Безкоштовно	Безкоштовно	Плата за ліцензію та обслуговування	Плата за ліцензію та обслуговування

Варто також розглянути сильні та слабкі сторони цих інструментів для тестування [7]. Нижче наведено короткий опис сильних сторін та обмежень інструментів (таб.2.2) на основі порівняння вище.

Таблиця 2.2

Порівняння сильних сторін та обмежень інструментаріїв

Інструменти	Сильні сторони	Обмеження
TestComplete	Багато мов сценаріїв на вибір. Потрібні лише базові навички програмування.	Для TestComplete необхідні значні збори за ліцензування та обслуговування.

Продовження таблиці 2.2

Katalon Studio	Жодної плата за ліцензування та обслуговування не вимагає. Інтеграція необхідних фреймворків та функцій для швидкого створення та виконання тестових кейсів. Побудований поверх фреймворку Selenium, але виключає необхідність у вдосконалених навичках програмування, необхідних для Selenium.	Рішення, яке виникає з невеликою спільнотою Набір функцій все ще змінюється. Відсутність варіантів для мов сценаріїв: підтримується лише Java / Groovy.
Selenium	Відкритий код, без ліцензії та плати за обслуговування Велика та активна спільнота розробників та користувачів, щоб не відставати від програмних технологій. Відкритий для інтеграції з іншими інструментами та структурами для розширення його можливостей	Тестувальні команди повинні мати хороші навички програмування та досвід для налаштування та інтеграції Selenium з іншими інструментами та структурами. Новим командам потрібно заздалегідь вкласти час для налаштування та інтеграції Повільна підтримка спільноти.
UFT	Комплексні функції автоматизованого тестування, інтегровані в єдину систему. Спеціальна підтримка користувачів плюс створена велика спільнота користувачів.	Дороге рішення: плата за ліцензію та обслуговування значно висока. Можливі великі витрати на оновлення та додаткові модулі.

В ході опитування спільноти тестувальників автоматизації платформу Selenium використовували понад 80% опитаних. Це не просто окремий інструмент, це повний пакет. Це набір інструментів, що складається з досить багатьох компонентів, кожен з яких відіграє чітку роль у розробці веб-додатків. Варто розглянути докладніше Selenium так як він включає в себе Selenium WebDriver, Selenium IDE та Selenium Grid, Selenium Remote Control [8].

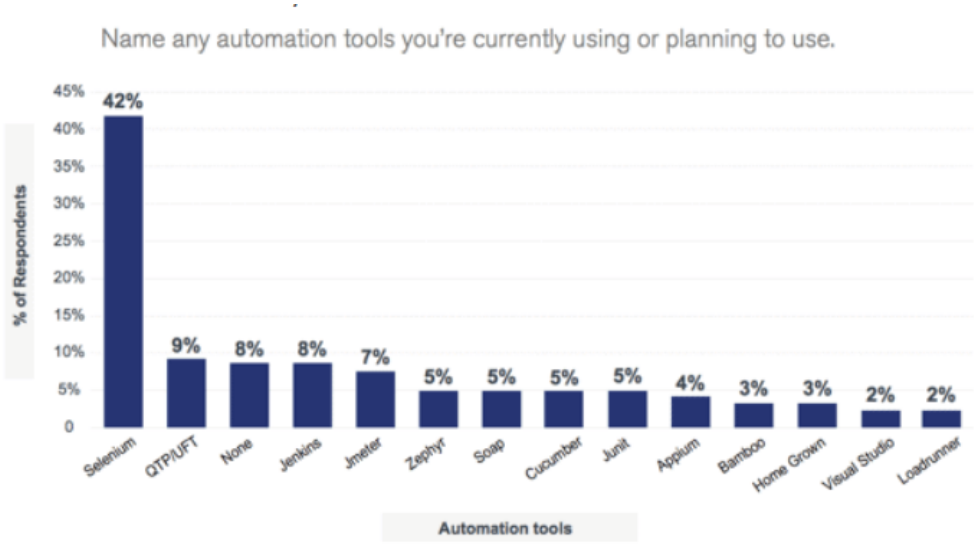


Рис. 2.2. Статистика використання автоматизованих інструментаріїв

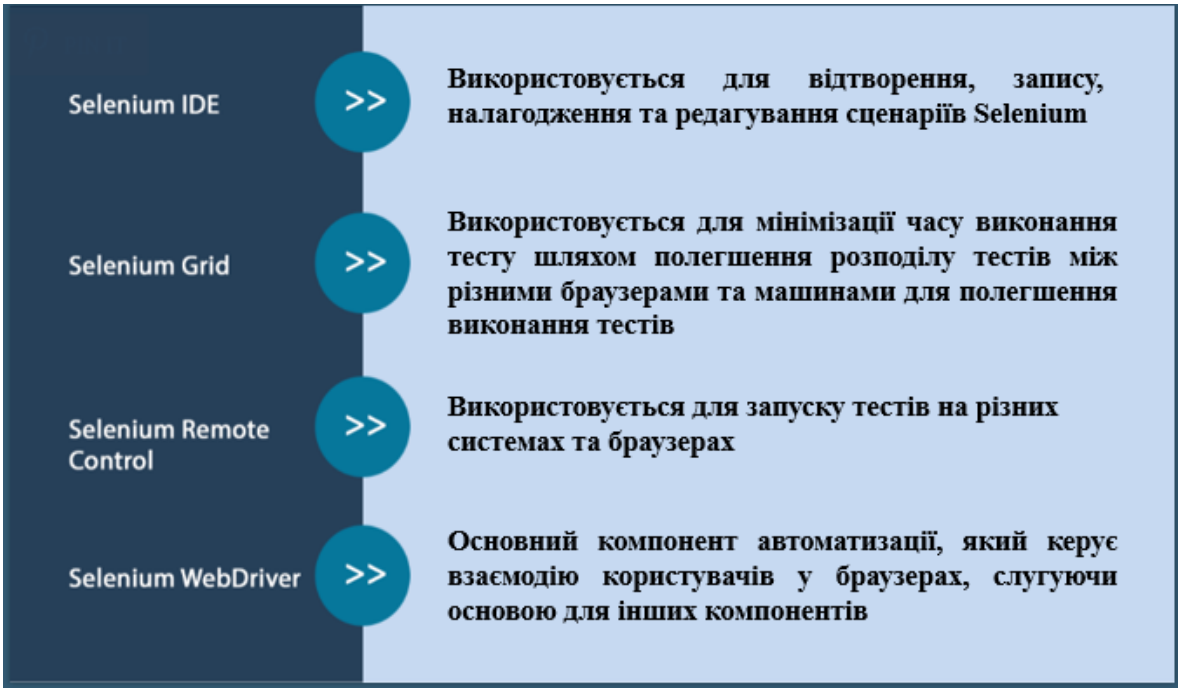


Рис. 2.3. Опис Selenium

- Selenium WebDriver - це основний компонент автоматизації, який керує взаємодією користувачів у браузерах, слугуючи основою для інших компонентів.

- Selenium IDE - це розширення браузера, яке записує та відтворює взаємодію користувачів у браузері.

- Selenium Grid дозволяє WebDriver запускати тести автоматизації на багатьох машинах, браузерах та операційних системах одночасно.

- Selenium Remote Control – це важливим компонент у наборі тестів на Selenium. Це рамка тестування, яка дозволяє службі контролю якості або розробнику писати тестові кейси будь-якою мовою програмування для автоматизації тестів інтерфейсу користувача для веб-додатків проти будь-якого веб-сайту HTTP [9].

2.5. Мова програмування Java

Java - це мова програмування загального призначення, яка належить корпорації Oracle. Java побудована на принципах об'єктно-орієнтованого програмування. Мова дотримується принципу WORA (Write Once, Run Anywhere), що приносить багато переваг між платформами.

Багато великих корпорацій використовують Java для обслуговування своїх внутрішніх систем. Java - найпоширеніша мова програмування, що використовується для автоматизації тестів. Величезні 44% клієнтів використовують Java для своїх автоматизованих перевірок. Існує безліч легко доступних фреймворків, плагінів та освітніх ресурсів, які підтримують Java для автоматизації тестів – що доводить, що підтримка спільноти є рушійним фактором при виборі мови програмування для автоматизації тестів. Якщо команди узгоджують свої засоби автоматизації тестів із інструментами їх розробки продуктів, це також може пояснити популярність Java для перевірок інтерфейсу користувача. Незважаючи на

те, що JUnit є популярною структурою модульного тестування, низка платформ тестування автоматизації з відкритим кодом розроблено за допомогою Java. Автоматизоване тестування браузера для веб-продукту (веб-сайту / веб-програми) можна виконати за допомогою JUnit разом із Selenium WebDriver [10].

2.6. Середовище для автоматизованого тестування інтернет ресурсів

IntelliJ IDEA має деякі найпродуктивніші функції заповнення коду Java. Його алгоритм прогнозування може точно припустити, що кодер намагається ввести, і заповнити його для нього, навіть якщо він не знає точної назви певного класу, учасника чи будь-якого іншого ресурсу.

IntelliJ IDEA - це інтегроване середовище розробки (IDE) для мов JVM (але численні плагіни можуть розширити його, щоб забезпечити поліглот), призначене для максимізації продуктивності розробників. Він виконує для вас рутинні та повторювані завдання, забезпечуючи розумне завершення коду, статичний аналіз коду та рефакторинг, а також дозволяє зосередитися на яскравій стороні розробки програмного забезпечення, роблячи це не тільки продуктивним, але і приємним досвідом. IntelliJ IDEA - це платформа IDE, яка забезпечує стабільний досвід роботи в Windows, macOS та Linux.

IntelliJ IDEA використовується для розробки програм наступними мовами, які можна скомпілювати в байт-код JVM, а саме: Java, Kotlin, Scala, Groovy. IntelliJ IDEA забезпечує середовище, орієнтоване на редактора. Він слідкує за вашим контекстом і автоматично пропонує необхідні інструменти, які допоможуть мінімізувати ризик переривання потоку розробника [11].

Висновки до розділу 2

В даному розділі були розглянуті найвідоміші і найбільш використовуваніші види тестування, а також способи тестування. На високому рівні нам потрібно розрізняти ручні та автоматизовані тести. Тестування вручну проводиться особисто,

натискаючи додаток або взаємодіючи із програмним забезпеченням та API за допомогою відповідних інструментів. Це дуже дорого, оскільки для цього потрібно, щоб хтось створив середовище та виконував тести самостійно, і це може спричинити людські помилки, оскільки тестувальник може робити помилки друку або пропускати кроки у сценарії тесту.

Автоматизовані тести, навпаки, виконуються машиною, яка виконує тестовий скрипт, який був написаний заздалегідь. Ці тести можуть сильно відрізнятися за складністю, від перевірки одного методу в класі до переконання, що виконання послідовності складних дій в інтерфейсі призводить до однакових результатів.

Також було обрано середовище розробки для написання тестів IntelliJ IDEA та мова Java, на якій будуть реалізовуватись тести. В ході виконання порівняння платформ, які будуть використовуватись для написання тесту, було обрано Selenium, так як він найпопулярніший серед розробників, відкритий код, без ліцензії та плати за обслуговування, а також відкритий для інтеграції з іншими інструментами та структурами для розширення його можливостей.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ АВТОТЕСТУ ЗА ДОПОМОГОЮ SELENIUM ДЛЯ INTERNET-МАГАЗИНУ «CANADIAN TIRE»

3.1. Створення тестового сценарію

Першим кроком ми розглянемо де використовується Selenium і як він прописується в коді. На рис. 3.1. Можна побачити прописаний WebDriver, який використовується для того, щоб можна було взаємодіяти з елементами сторінок та описувати очікування WebDriverWait.

Розглянемо клас BasePage, де і відбувається оголошення веб драйвера та вейтерів.

```
public class BasePage {

    protected WebDriver driver;

    public BasePage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, page: this);
    }

    public void waitForPageLoadComplete(long timeToWait) {
        new WebDriverWait(driver, timeToWait).until(
                webDriver -> ((JavascriptExecutor) webDriver).executeScript( s: "return document.readyState").equals("complete"));
    }

    public void waitForAjaxToComplete(long timeToWait) {
        new WebDriverWait(driver, timeToWait).until(
                webDriver -> ((JavascriptExecutor) webDriver).executeScript( s: "return window.jQuery != undefined && jQuery.active == 0;"));
    }

    public void waitForAjaxToCompletePdp(long timeToWait) {
        new WebDriverWait(driver, timeToWait).until(
                webDriver -> ((JavascriptExecutor) webDriver).executeScript( s: "return window.jQuery != undefined && jQuery.active <=2;"));
    }

    public void waitVisibilityOfElement(long timeToWait, WebElement element) {
        WebDriverWait wait = new WebDriverWait(driver, timeToWait);
        wait.until(ExpectedConditions.visibilityOf(element));
    }
}
```

Рис. 3.1. Клас BasePage: оголошення веб драйвера та вейтерів

Щоб усе це працювало необхідно було б колись скачувати актуальну версію драйвера, кидати його в папку з проектом та підв’язувати його у цьому класі. Але нині технології пішли вперед і ці всі дії ми можемо замінити цими стрічками у POM файлі

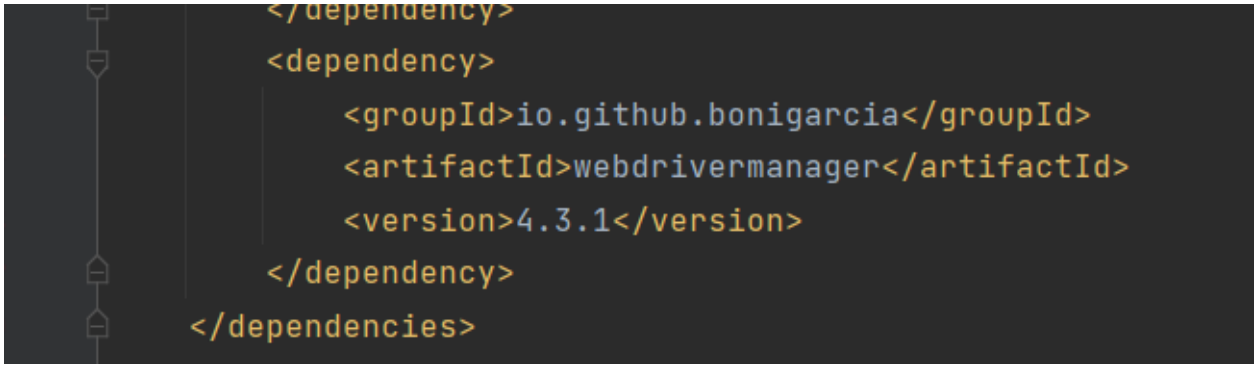


Рис. 3.2. Опис автоматизованого підключення Webdriver через POM файл

Це дозволяє автоматично отримувати актуальну версію веб-драйвера та взаємодіяти з сторінкою без додаткових зусиль.

Тепер, коли веб-драйвер встановлений, варто розглянути, які дії потрібно виконувати далі. Нині популярним патерном є PFM, тому і будемо його притримуватися. Що ж він робить і чим може нам допомогти?

Завд’яки йому весь функціонал та всі елементи сторінок, з якими ми взаємодіємо співвідносяться до аналогічних класів-сторінок. Там відбувається вся взаємодія (в методах) та пошук чи очування потрібних елементів. Приведемо приклади наших класів сторінок.

```java
public class CheckoutPage extends BasePage {

    @FindBy(xpath = "//button[contains(@class,'choose-payment-method__btn--regular-card')]")
    private WebElement paymentCartButton;

    @FindBy(xpath = "//*[@class='opc-billing-form']/*[@class='opc-billing-form__wrapper']")
    private WebElement billingForm;

    @FindBy(xpath = "//div[@class='checkout-payment-form__wrapper']")
    private WebElement paymentForm;

    @FindBy(xpath = "//button[contains(@class,'checkout-order-summary__continue-btn')]")
    private WebElement completeOrderButton;

    public CheckoutPage(WebDriver driver) { super(driver); }

    public void clickPaymentCartButton() { paymentCartButton.click(); }

    public boolean isBillingFormVisible() { return billingForm.isDisplayed(); }

    public boolean isPaymentFormVisible() { return paymentForm.isDisplayed(); }

    public boolean isCompleteOrderButtonVisible() { return completeOrderButton.isDisplayed(); }
}
```

Рис. 3.3. Клас Checkout Page

```java
public class HomePage extends BasePage {

    @FindBy(xpath = "//header[contains(@class, 'global-header global-header--sticky') or @class='page-header']")
    private WebElement header;

    @FindBy(xpath = "//footer")
    private WebElement footer;

    @FindBy(xpath = ".//a[@class='global-header__main-bar__utility-nav__user-cart__link']")
    private WebElement cartIcon;

    @FindBy(xpath = ".//a[contains(@class, 'header-top-bar__input__language')]/span")
    private WebElement languageButton;

    @FindBy(xpath = "//button[contains(@class,'enterprise-account__button_sign-in')]")
    private WebElement signInButton;

    @FindBy(xpath = "//button[contains(@class, 'enterprise-account__button_register')]")
    private WebElement registerButton;

    @FindBy(xpath = "//div[@class='gigya-screen-dialog-main']")
    private WebElement signInPopup;

    @FindBy(xpath = ".//input[@name='username'][@placeholder='Email *']")
    private WebElement emailField;
```

Рис. 3.4. Клас HomePage: опис локаторів

Локатор - це команда, яка повідомляє Selenium, які елементи графічного інтерфейсу (скажімо, текстове поле, кнопки, прапорці тощо) потрібні для роботи.

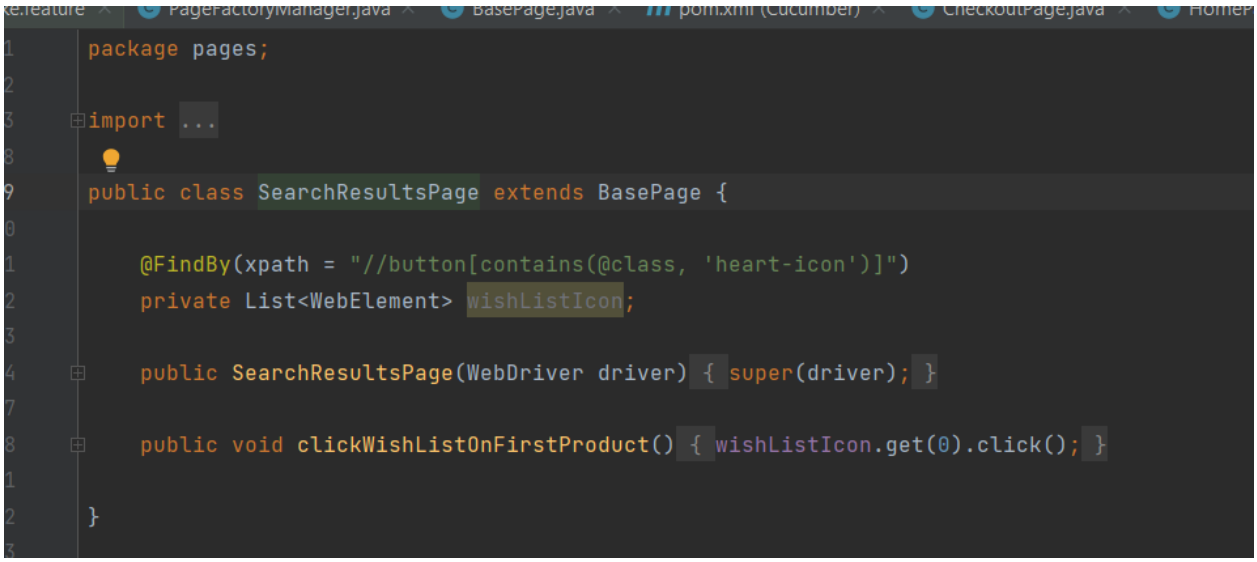
Визначення правильних елементів графічного інтерфейсу є необхідною умовою для створення сценарію автоматизації.

```
98
99      public WebElement getSignInPopup() { return signInPopup; }
102
103     public void clickSignInPopupCloseButton() {
104         ((JavascriptExecutor) driver).executeScript( s: "arguments[0].click()", signInPopupCloseButton);
105     }
106
107     public void clickStoreButton() { storeButton.click(); }
110
111     public boolean isStorePopupVisible() { return storePopup.isDisplayed(); }
114
115     public void isSearchFieldVisible() { searchField.isDisplayed(); }
118
119     public void clickCartButton() { cartIcon.click(); }
122
123     public void clickLanguageButton() { languageButton.click(); }
126
127     public void enterTextToSearchField(final String searchText) {
128         searchField.clear();
129         searchField.sendKeys(searchText);
130     }
131
132     public void clickSearchButton() { searchButton.click(); }
135
136     public WebElement getWishListProductsCount() { return wishListProductsCount; }
139
140     public String getAmountOfProductsInWishList() { return wishListProductsCount.getText(); }
143
144 }
```

Рис. 3.5. Клас HomePage: опис методів

```
public class ProductPage extends BasePage {

    @FindBy(xpath = "//div[@class='add-to-cart__button-wrapper']/button[contains(@class,'add-to-cart__button')]")
    private WebElement addToCartButton;

    @FindBy(xpath = "//div[@class='success-popup__shopping-wrapper']//h3[@class='success-popup__success-message']")
    private WebElement addToCartPopupHeader;

    @FindBy(xpath = "//a[contains(text(),'Continue shopping')]")
    private WebElement continueShoppingButton;

    @FindBy(xpath = "//a[contains(text(),'Continue to cart')]")
    private WebElement continueToCartButton;

    public ProductPage(WebDriver driver) { super(driver); }

    public void clickAddToCartButton() {
        ((JavascriptExecutor) driver).executeScript( s: "arguments[0].click();", addToCartButton);
    }

    public boolean isAddToCartPopupVisible() { return addToCartPopupHeader.isDisplayed(); }

    public void isContinueShoppingButtonVisible() { continueShoppingButton.isDisplayed(); }

    public String getAddToCartPopupHeaderText() { return addToCartPopupHeader.getText(); }

    public void isContinueToCartButtonVisible() { continueToCartButton.isDisplayed(); }
```

Рис. 3.6. Клас ProductPage: опис методів і локаторів



```java
package pages;

import ...

public class SearchResultsPage extends BasePage {

    @FindBy(xpath = "//button[contains(@class, 'heart-icon')]")
    private List<WebElement> wishListIcon;

    public SearchResultsPage(WebDriver driver) { super(driver); }

    public void clickWishListOnFirstProduct() { wishListIcon.get(0).click(); }

}
```

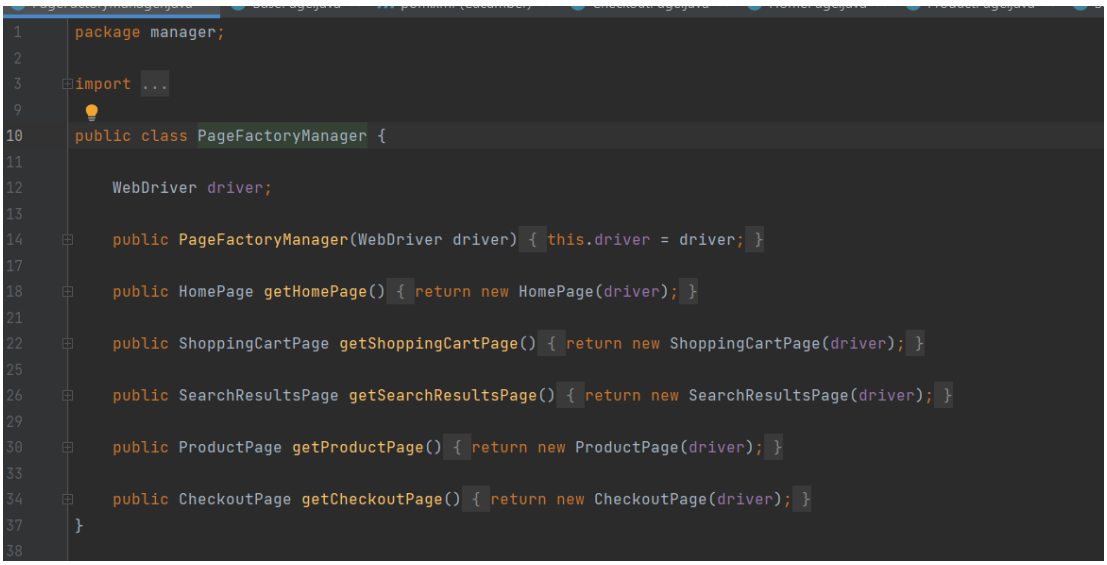
Рис. 3.7. Клас SearchResultPage

```java
package pages;

import ...

public class ShoppingCartPage extends BasePage {

    @FindBy(xpath = "//h1[@class='checkout-header__heading']")
    private WebElement shoppingCartTitle;

    @FindBy(xpath = "//button[@class='checkout-order-summary__continue-btn']")
    private WebElement checkoutButton;

    @FindBy(xpath = "//div[contains(@class, 'shopping-cart-item--shopping-cart-your-order')]|//section[@data-code or @data-product-code]")
    private WebElement shoppingCartItem;

    public ShoppingCartPage(WebDriver driver) { super(driver); }

    public WebElement getShoppingCartTitle() { return shoppingCartTitle; }

    public boolean isShoppingCartTitleVisible() { return shoppingCartTitle.isDisplayed(); }

    public void clickCheckoutButton() { checkoutButton.click(); }

    public WebElement getShoppingCartItem() { return shoppingCartItem; }
```

Рис. 3.8. Клас ShoppingCartPage

Можна помітити, що часто використовується аннотація Find by, що є просто кричущою ознакою нашого патерна Page Factory.

Тепер коли ми маємо заготовки наших класів потрібно мати спосіб отримувати їх конкретні екземпляри та взаємодіяти з ними. У цьому нам допоможе спеціальний клас PageFactoryManager.



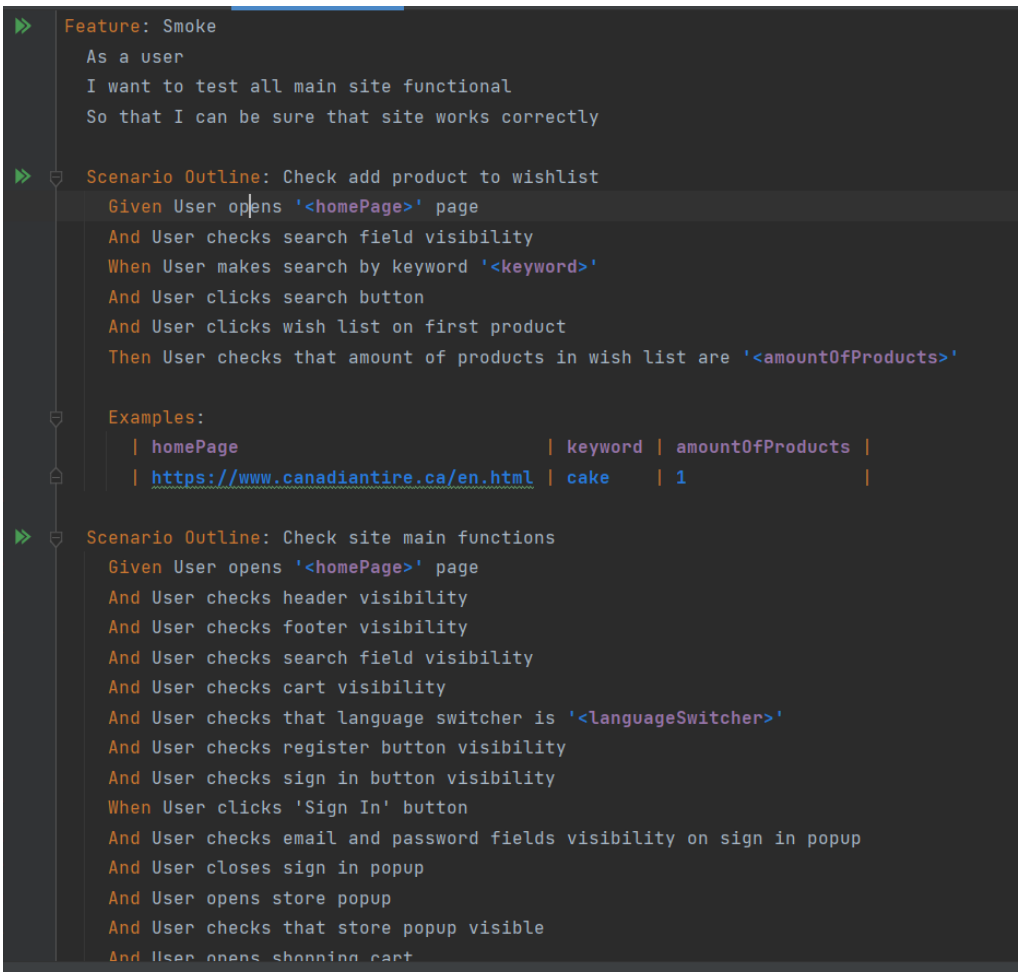
```
package manager;

import ...

public class PageFactoryManager {

    WebDriver driver;

    public PageFactoryManager(WebDriver driver) { this.driver = driver; }

    public HomePage getHomePage() { return new HomePage(driver); }

    public ShoppingCartPage getShoppingCartPage() { return new ShoppingCartPage(driver); }

    public SearchResultsPage getSearchResultsPage() { return new SearchResultsPage(driver); }

    public ProductPage getProductPage() { return new ProductPage(driver); }

    public CheckoutPage getCheckoutPage() { return new CheckoutPage(driver); }
}
```

Рис. 3.9. Клас PageFactoryManager

Даний клас буде використовуватися як білдер, щоб повертати нам готові еземпляри бажаних сторінок, в них уже будемо викладати потрібні нам методи і взаємодіяти з сторінкою. Це все можна побачити з методів нашого класу-менеджеру.

Тепер коли у нас є готові сторінки з потрібним нам функціоналом та спосіб повертати їх, ми можемо приступити до побудови самих тестів технологією BDD. Існує два способи зробити це – перший це відразу почати написання логіки тестів у .feature файлі а вже потім описати кроки за допомогою методів наших сторінок у спеціальному класі визначнику, або ж спершу описати кроки у класі, а потім по них створити .feature файл. Мені більш зручний 2 спосіб, оскільки ми не втрачаємо загального огляду картини, вдавшись в деталі. Отже розглянемо перший спосіб.

Створимо Featiure file.



```
Feature: Smoke
  As a user
  I want to test all main site functional
  So that I can be sure that site works correctly

  Scenario Outline: Check add product to wishlist
    Given User opens '<homePage>' page
    And User checks search field visibility
    When User makes search by keyword '<keyword>'
    And User clicks search button
    And User clicks wish list on first product
    Then User checks that amount of products in wish list are '<amountOfProducts>'

    Examples:
      | homePage                          | keyword | amountOfProducts |
      | https://www.canadiantire.ca/en.html | cake    | 1                |

  Scenario Outline: Check site main functions
    Given User opens '<homePage>' page
    And User checks header visibility
    And User checks footer visibility
    And User checks search field visibility
    And User checks cart visibility
    And User checks that language switcher is '<languageSwitcher>'
    And User checks register button visibility
    And User checks sign in button visibility
    When User clicks 'Sign In' button
    And User checks email and password fields visibility on sign in popup
    And User closes sign in popup
    And User opens store popup
    And User checks that store popup visible
    And User opens shopping cart
```

Рис. 3.10. Feature File

Тут ми використовуємо ключові слова технології BDD, які ви можете бачити на рис. 3.10. Коли фіча файл створений, то варто підв'язати наші кроки до сторінок створивши додатковий клас з визначенням кроків.

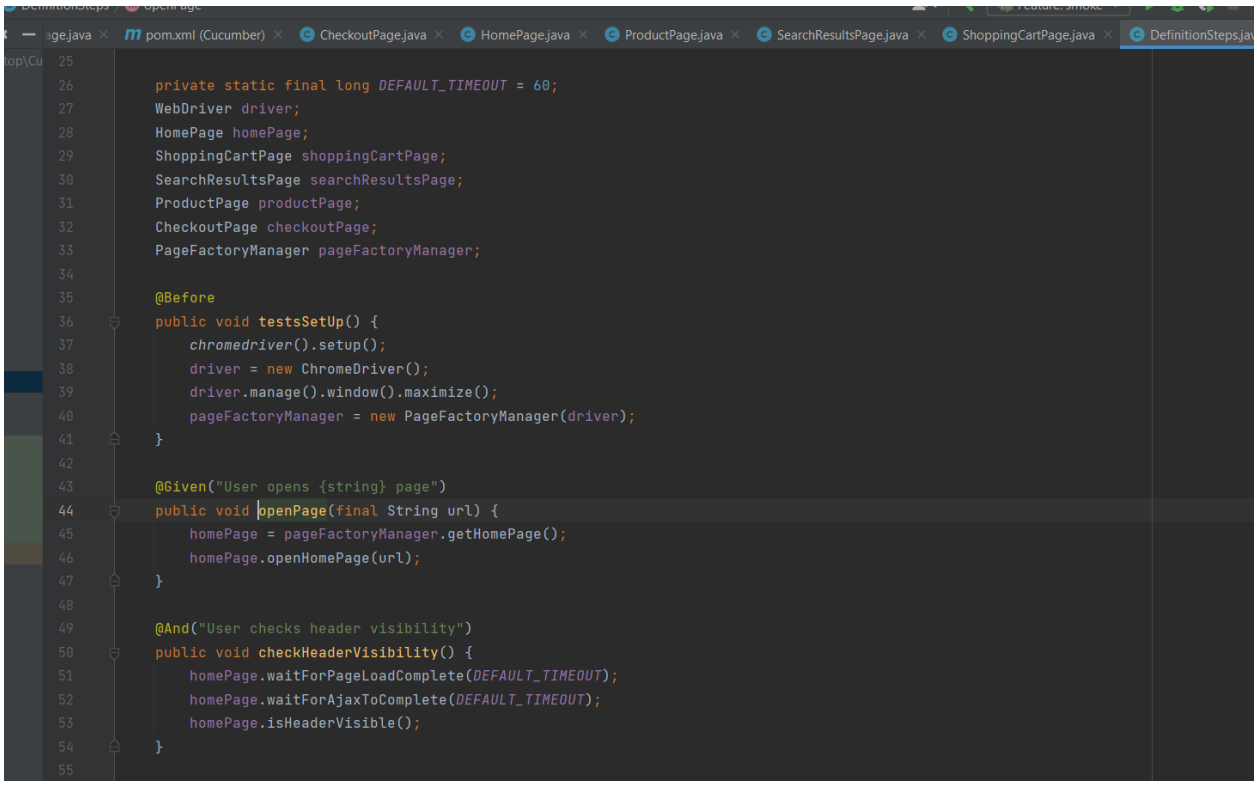


Рис. 3.11. Клас з реалізацією кроків

Ось наш клас і у ньому анотації у останніх двох методах цілком співпадають по назві з кроками у фіча файлі. Дуже гарним способом вважається просто натиснути на створеному кроці у фіча-файлі правою кнопкою миші та вибрати «створити нове визначення кроку»

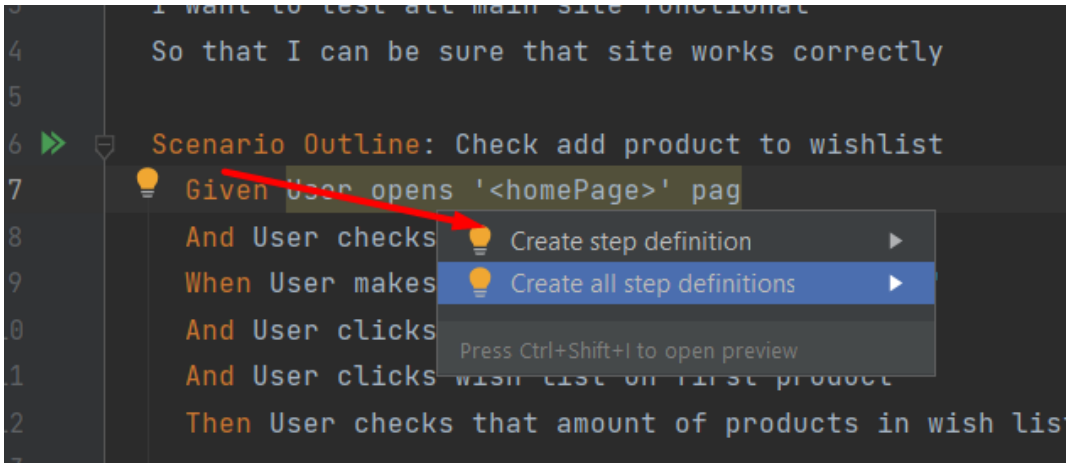


Рис. 3.13. Створення нового визначення кроку

Після цього буде запропоновано вибрати клас з крокампи потім у цьому класі створиться метод з анотацією ідентичною нашому крокові у фіча-файлі. Це рекомендується робити для того, щоб уникнути помилок.

Тепер увесь код написантй і все, що залишається виконати - це запустити автотести і переконатися, що все працює так як потрібно.

3.2. Запуск автотесту для Internet-магазину «Canadian Tire»

Для того щоб запустити тести потрібно мати браузер, завантажений проект та середовище розробки Intellij IDEA для приємного графічного інтерфейсу.

Ми відкриваємо проект в Intellij IDEA натиснувши відповідну кнопку у нашому середовищі: File ->Open.

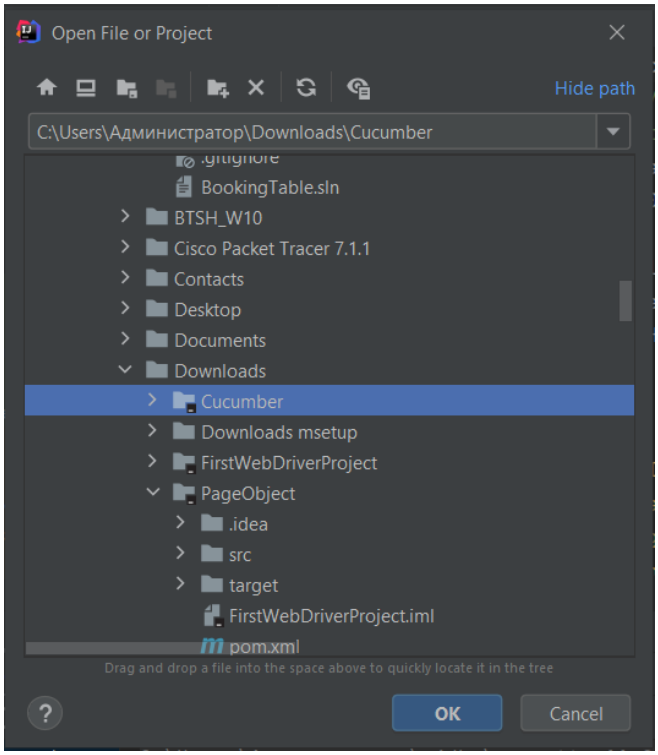


Рис. 3.12. Вікно вибору проекту

Потім переходимо у фіча-файл та запускаємо всі сценарії (набір тестів), натиснувши на «зелені трикутники» як зображенно на рис. 3.13.

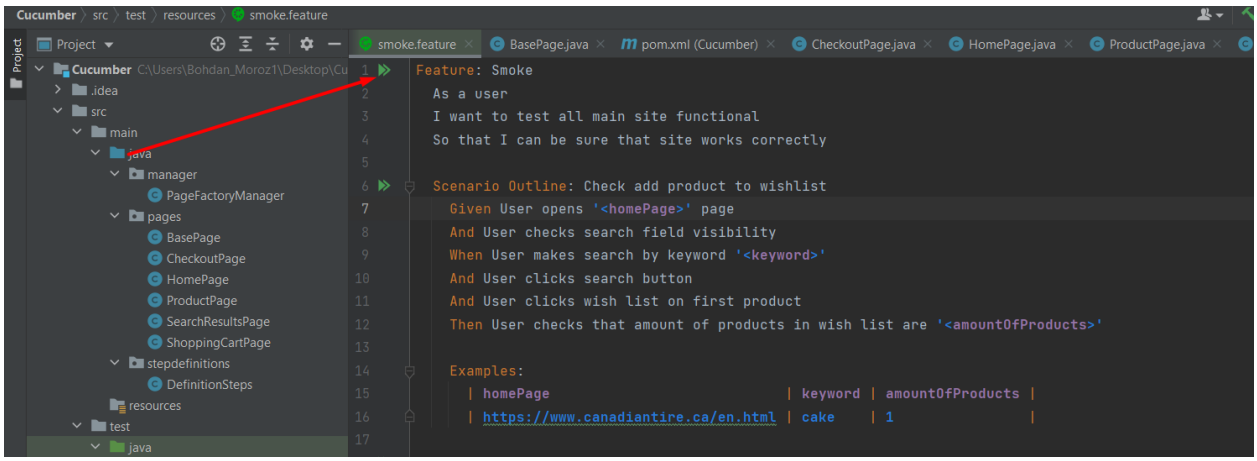


Рис. 3.13. Запуск всіх сценаріїв

А якщо потрібно запустрити тільки один певний тест, то він запускється біля Scenario.

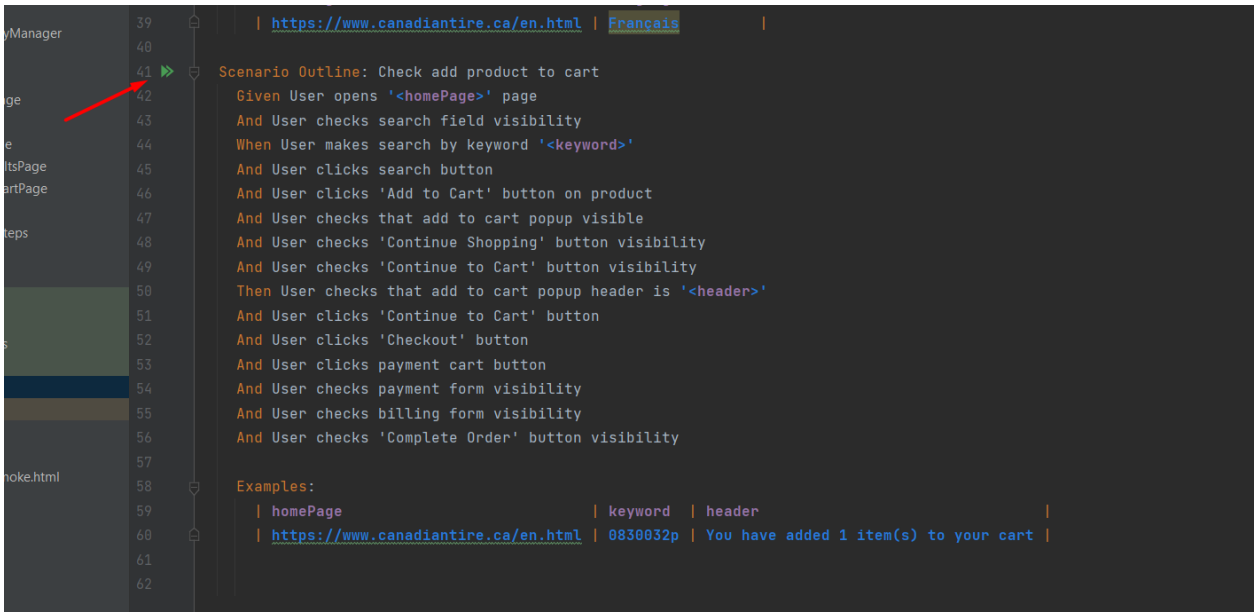


Рис. 3.14. Запуск одного сценарію

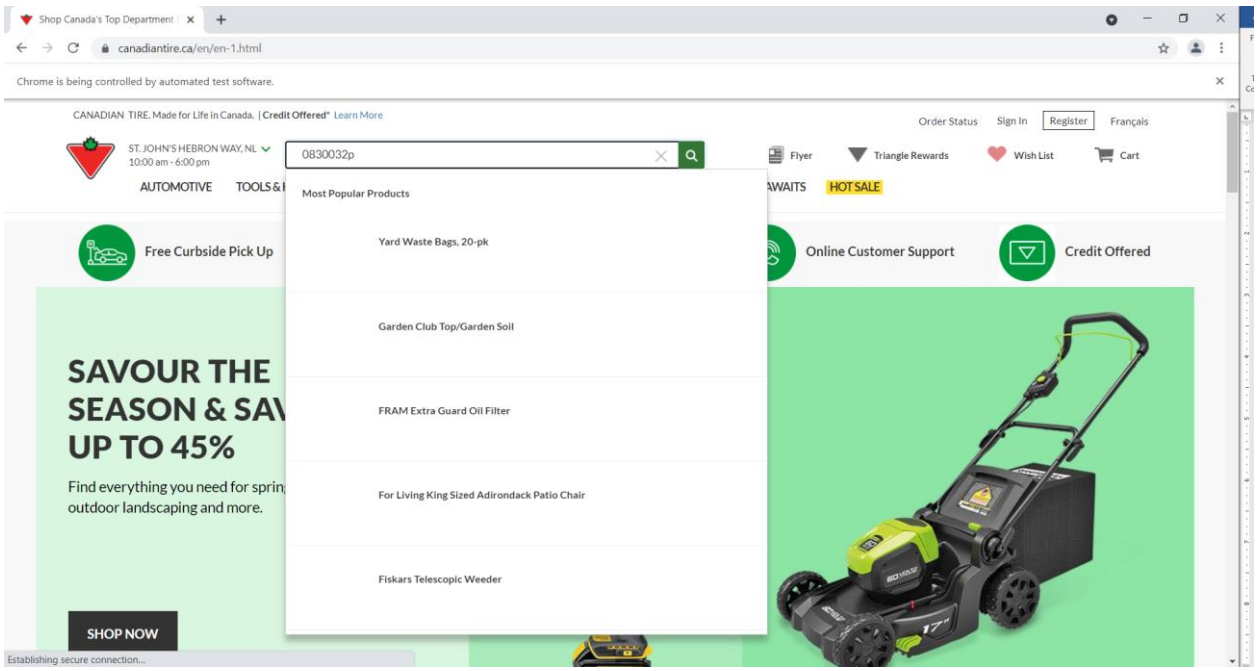


Рис. 3.15. Виконання тестів

Перевіряємо чи його результат є успішним. В даному випадку тести пройшли успішно.

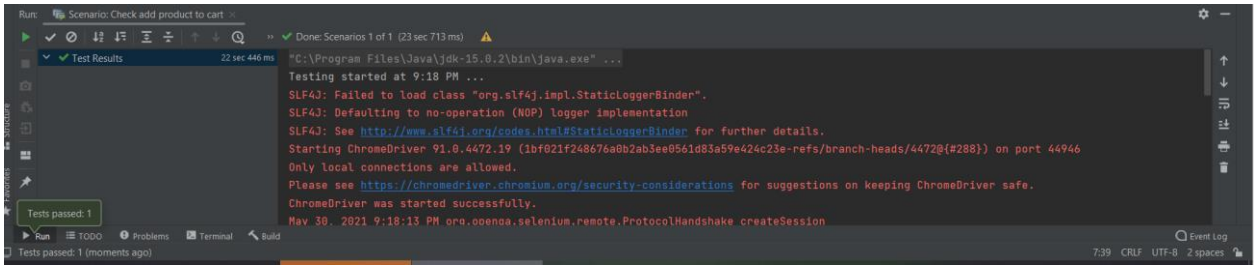


Рис. 3.16. Повідомлення про успішно пройдені тести

Якщо ж ні то шукаємо проблему в стеку викликів або дебажимо наш сценарій за допомогою Debug.

Після цього ми можемо сформувати звіт, який буде у графічно доречній формі та буде відкриватися у будь-якому браузері. Виконання даної дії щображено на рис.

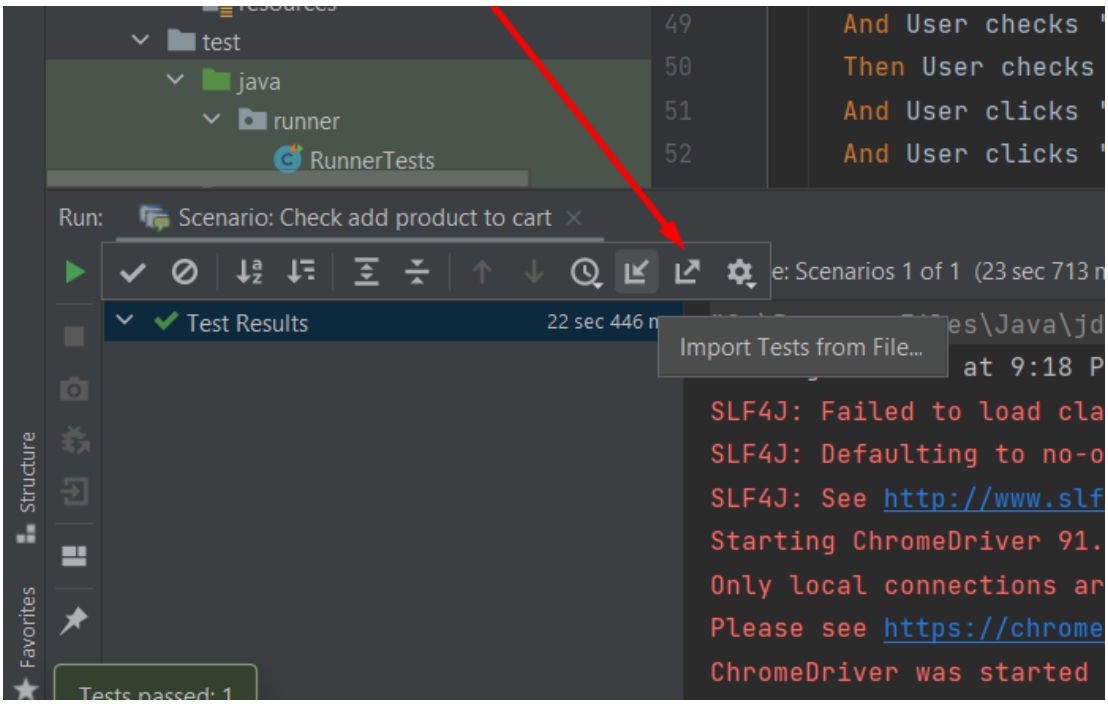


Рис. 3.17. Імпорт тестів з файлу

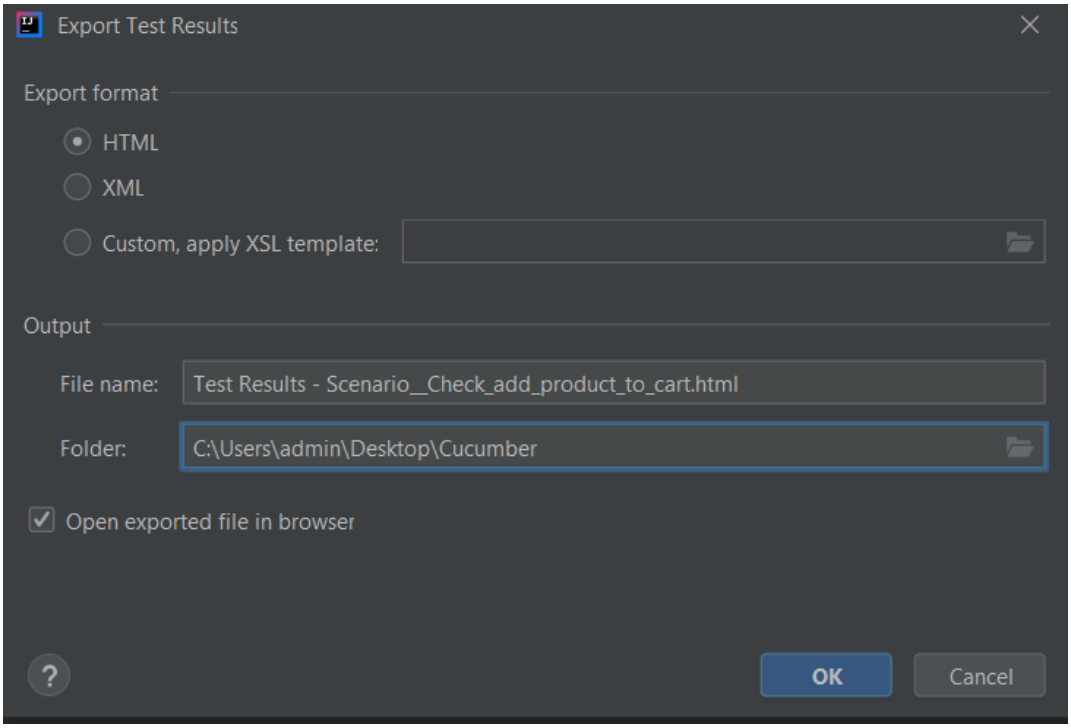


Рис. 3.18. Вибір формату та розміщення звіту

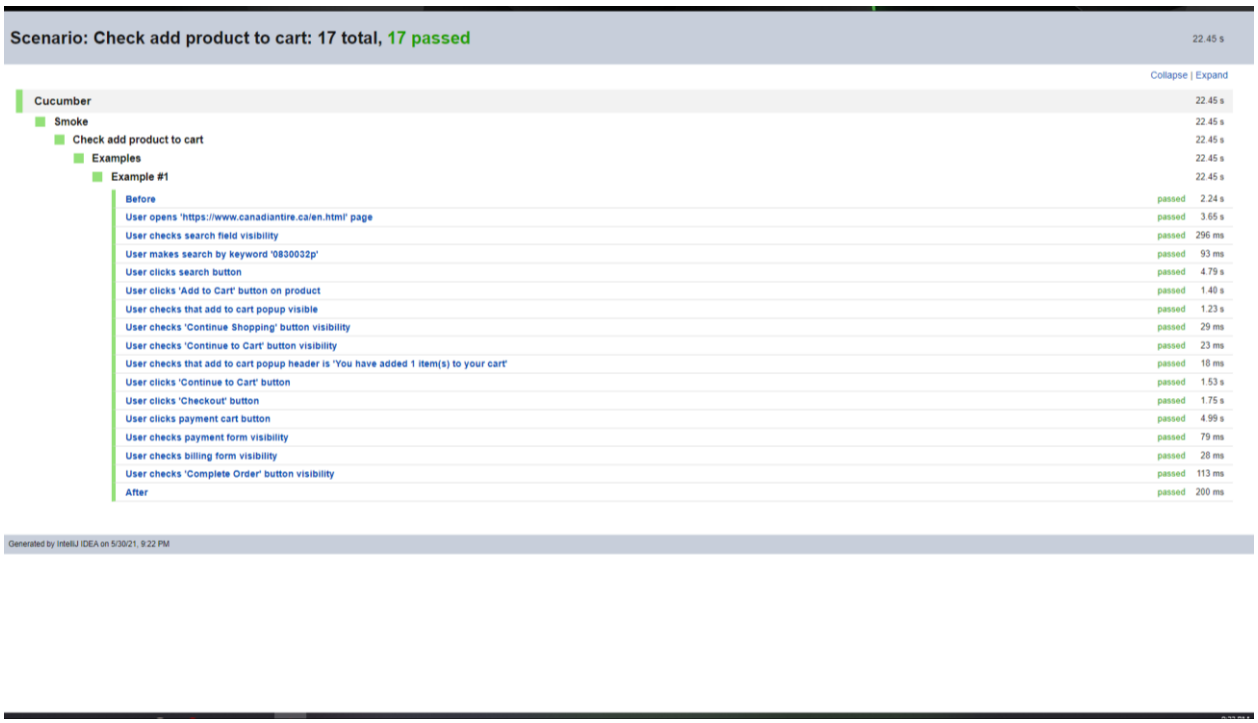


Рис. 3.19. Сформований звіт у браузері

Висновки до розділу 3

В даному розділі було створенно в середовищі Intellij IDEA та виконано Smoke тестування основного функціоналу сайту «CANADIAN TIRE» за допомогою платформи Selenium та було використано інтеграцію компонентів Maven, Junit, Cucumber, PFM. Також було запущенно тести, які у висновку нам видали успішний результат, тобто помилок не було знайдено.

ВИСНОВКИ

У дипломному проекті було досліджено всі види інтернет-ресурсів, всі види та способи тестування, а також було порівняно інструментарії для автоматизованого тестування.

Для виконання цілі був проведений аналіз видів інтернет-ресурсів, які важливі для користувачів та обрано для тестування вид ресурсу, який потребує більш досконалого тестування, аніж інші, а також аналіз видів тестування, та вибір виду, який охоплює весь основний функціонал сайт. Було порівняно та обрано найкращу платформу для виконання даного завдання.

Так як кожен споживач, особливо у період пандемії, користувався коменрційними сайтами для покупки певних товарів, то було обрано комерційний веб сайт «CANADIAN TIRE». Дуже важливо для користувачів, щоб працював основний функціонал сайту: замовлення, оплата, перегляд товарів, вибір фільтрів, пошук. Проаналізувавши всі способи та види тестування було обрано автоматизувати Smoke тестування, так як для ручного тестування смоук тестування буде відбуватися занадто довго, також людина може пропустити помилку через людський фактор, і коштувати також буде дорого, на відміну від автоматизованого. Але автоматизація тестів це дуже дорого, тому не всі тести можна атоматизувати, тому автоматизація це лише доповнення до мануального тестування. Для виконання автотесті звичайно потрібно інструментарій, для цього було обрано платформу Selenium, яка було обрана, шляхом порівняння інших засобів. Також для тестування було використано інтеграцію компонентів Maven, Junit, Selenium, Cucumber, PFM. Для реалізації тестів була обрана мова Java та середовище розробки IntelliJ IDEA, так як автотести зручніше писати на даній мові, бо вона є кросплатформенною та в даному середовищі, яке найкраще підходить для написання тестів на даній мові і дане середовище слідкує за контекстом і автоматично пропонує необхідні інструменти, які допоможуть мінімізувати ризик переривання потоку розробника. Дані тести можна використовувати для будь-якого комерційного сайту, замінивши лише локатори та URL адресу.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Internet Resources definition [Електронний ресурс] - 2021. - Режим доступу: https://www.lawinsider.com/ (дата звернення 10.05.2021 р.). - Назва з екрана.
2. 20 different types of websites: Part 1[Електронний ресурс] – Режим доступу: https://www.gwsmedia.com/ (дата звернення 11.05.2021 р.). - Назва з екрана.
3. Types of software testing [Електронний ресурс] - 2017. – Режим доступу: https://geteasyqa.com/ (дата звернення 12.05.2021 р.). - Назва з екрана.
4. The different types of software testing [Електронний ресурс] – 2021. – Режим доступу: https://www.atlassian.com/ (дата звернення 12.05.2021 р.). - Назва з екрана.
5. Types of software testing: Different testing types with details [Електронний ресурс] – 2021. – Режим доступу: https://www.softwaretestinghelp.com/ (дата звернення 12.05.2021 р.). - Назва з екрана.
6. Manual testing vs Automation testing: How much dies it really cost? [Електронний ресурс] – 2020. – Режим доступу: https://medium.com/ (дата звернення 15.05.2021 р.). - Назва з екрана.
7. A comparison of Automated testing tools [Електронний ресурс] – 2017. - Режим доступу до ресурсу: https://dzone.com/ (дата звернення 15.05.2021). - Назва з екрана.
8. The good and the bad of Selenium test automation software [Електронний ресурс] – 2021. – Режим доступу: https://www.altexsoft.com/ (дата звернення 16.05.2021). - Назва з екрана.
9. Selenium RC: Differences from Webdriver [Електронний ресурс] – 2019. – Режим доступу: https://www.browserstack.com/ (дата звернення 17.05.2021). - Назва з екрана.
10. Which programming language is most popular for UI test automation in 2019? [Електронний ресурс] – Режим доступу: https://applitools.com/ (дата звернення 17.05.2021 р.). - Назва з екрана.

11. IntelliJ IDEA overview [Електронний ресурс] – 2021. – Режим доступу: https://www.jetbrains.com/ (дата звернення 17.05.2021 р.). - Назва з екрана.

12. Canadian Tire [Електронний ресурс] – Режим доступу: https://www.canadiantire.ca/ (дата звернення 18.05.2021 р.). - Назва з екрана.

ДОДАТКИ

Додаток А

Опис проекту в файлі POM (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns=http://maven.apache.org/POM/4.0.0
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance\
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>Cucumber</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
```

```
                <source>11</source>
                <target>11</target>
            </configuration>
        </plugin>
    </plugins>
</build>

<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
    </dependency>
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>6.8.1</version>
    </dependency>
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>6.8.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
    </dependency>
    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>4.3.1</version>
    </dependency>
</dependencies>

</project>
```