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

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри
_____ А.С. Савченко
«____» _____ 20___р

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА)

ВИПУСКНИКА ОСВІТНЬОГО СПУПЕНЯ «БАКАЛАВР»

Тема: «Система тестування вмісту веб-сайтів»

Виконавець: студент УС-411 Мороз Богдан Петрович

(студент, група, прізвище, ім'я, по батькові)

Керівник: к. т. н., доцент Колісник Олена Василівна

(науковій ступень, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: ст. викл. Шевченко О.П.

(П.І.Б.) (підпис)

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
А.С. Савченко
“_____”_____ 2021 р.

ЗАВДАННЯ
на виконання дипломного проекту студента
Мороз Богдан Петрович

1. Тема: *Система тестування вмісту веб-сайтів* затверджена наказом ректора від 22.04.2021 № 636/ст.

2. Термін виконання: з 11.05.2021 р. по 14.06.2021 р.

3. Вихідні дані: будова та функціонування веб-сайтів; внутрішня структура веб-сайтів; сучасні приклади поширених технологій створення веб-сайтів; існуючі методи тестування наповненості веб-сайтів; базуючись на проведених дослідженнях сформулювати програмне рішення, та імплементувати його програмно у вигляді програмного рішення направлених на тестування наповненості контекстом веб-сайтів.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): Аналіз функціонування веб-сайтів. Аналіз внутрішньої структури веб-сторінок . Огляд існуючих програмних рішень тестування наповненості веб-сайтів. Розробка та дослідження програмного модуля тестування наповненості контентом веб-сайтів.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	12.05.2021	Виконано
2.	Аналіз літературних джерел	12.05.2021	Виконано
3.	Обґрунтування рішення	14.05.2021	Виконано
4.	Збір інформації	15.05.2021	Виконано
5.	Аналіз будови та функціонування веб-сторінки	17.05.2021	Виконано
6.	Аналіз існуючих підходів до тестування веб-сайту	18.05.2021	Виконано
7.	Розробка архітектури програмного модуля тестування веб-сторінки	20.05.2021	Виконано
8.	Розробка програмного комплексу тестування веб-сторінки	25.05.2021	Виконано
9.	Оформлення і друк пояснювальної записки	30.05.2021	Виконано
10.	Оформлення презентації	04.05.2021	Виконано
11.	Отримання рецензій від опонентів	09.06.2021	Виконано
12.	Підготовка до захисту в ЕК	11.06.2021 - 14.06.2021	Виконано

Дипломник _____Б.П. Мороз

(підпис, дата)

Дипломний керівник _____О.В Колісник

(підпис, дата)

РЕФЕРАТ

Дипломний проект складається зі вступу, трьох розділів, загальних висновків, списку бібліографічних використаних джерел, додатків, загальним обсягом робота складає 66 сторінок, має 22 рисунки, 2 сторінки додатків. Список бібліографічних посилань використаних джерел містить 21 найменування і займає 2 сторінки.

Метою дипломного проекту є створення програмного застосунку для тестування наповненості контентом веб-сторінок та їх базового функціоналу.

Об'єкт дослідження: процеси тестування наповненості контентом веб-сторінок та їх базового функціоналу.

Результат дослідження: тестовий фреймворк з можливістю тестування веб-сторінок.

Методи дослідження: математичний аналіз, системне проектування.

Предметом дослідження являється модель програмного рішення для тестування наповненості контентом веб-сторінок та їх базового функціоналу.

ВЕБ-СТОРІНКА, КОНТЕНТ, ТЕСТУВАННЯ НАПОВНЕНОСТІ, ПРОГРАМНІ МОДУЛІ ТЕСТУВАННЯ, АВТОМАТИЧНЕ ТЕСТУВАННЯ, ,UI, ЯКІСТЬ,JAVA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.

API - Прикладний програмний інтерфейс

HTML - Мова гіпертекстової розмітки

IDE - Інтегроване середовище розробки

SDLC - Життєвий цикл програмного забезпечення

BDD - Процес розробки програмного забезпечення

TDD - Процес розробки програмного забезпечення

FDD - Процес розробки програмного забезпечення

PDD - Процес розробки програмного забезпечення

ВСТУП

Актуальність. Веб-сторінки відіграють велику роль у нашому повсякденному житті. У розробці веб-сторінок використовуються різноманітні технології та підходи, які невпинно розвиваються та проходять апгрейд. Проте не завжди те що сучасніше є простішим і зрозумілішим. Чим складніша сторінка, тим легше допустити в ній помилку, яка вплине на базовий функціонал чи наповненість контекстом. Це у свою чергу негативно відобразиться на якості продукту та задоволеності ним користувачами. Тож, потрібно мати систему, яка проконтролює базовий функціонал та наповненість контекстом сайту та здійме тривогу у разі проблем.

В тому випадку, якщо сторінка має вразливості на базовому рівні, то це дозволяє зловмисникам і підриває безпеку даних користувачів чи власника. Також використовуючи ці вразливості зловмисник може проникнути у внутрішню мережу компанії.

Щодня у веб-сторінках знаходять масу різноманітних проблем на базовому рівні і ці помилки та їх виправлення коштують компанії-власнику шалених грошей. Щоб цьому запобігти потрібно створювати системи тестування, що і є загальноприйнятою практикою, та постійно їх модернізувати. В майбутньому це збереже значні кошти та репутацію компанії власника. Також гарним тоном рахується поєднання ручного тестування та автоматизованих систем. Ця комбінація дасть найбільшу ймовірність знайти помилку та виправити її потративши найменші кошти.

Хоча на ринку існує безліч інструментів для тестування веб-сторінок, завдяки яким можна проводити тестування різноманітних вразливостей, але не існує єдиного цілісного програмного рішення для комплексного тестування наповненості

контентом веб-сторінок та їх базового функціоналу. Створений програмний модуль вирішує це питання.

У першому розділі проведено аналіз структури та базового функціонування веб-сторінок. Для повного розуміння як повинен функціонувати програмний модуль тестування, необхідно вивчити область та випадки, де буде застосовуватись даний програмний продукт. Також необхідно розуміти, що саме буде тестуватись, для цього був проведений аналіз існуючих проблем з наповненістю контентом веб-сторінок та їх базового функціоналу. Для створення нового, більш ефективного та оптимального рішення було проаналізовано ринок існуючих програмних рішень, щоб зрозуміти недоліки наявних систем тестування.

У другому розділі здійснено проектування та розробка програмного модуля тестування наповненості контентом веб-сторінок та їх базового функціоналу. Програмне рішення розроблене на базі багаторівневої архітектури і являє собою інфраструктуру програмних рішень створених на мові програмування Java з використанням сучасних бібліотек та компонентів для комплексного тестування веб-додатків.

У третьому розділі було здійснено опис використання самого програмного продукту та наведено графічні ілюстрації використання усіх його модулів. Також було сформовано інструкцію вирішення найпоширеніших проблем, які можуть виникати під час експлуатації.

РОЗДІЛ 1
РОЗГЛЯД ОСНОВНИХ СКЛАДОВИХ ВЕБ-СТОРІНКИ ТА ЗАСОБІВ ЇЇ ТЕСТУВАННЯ

Сайт або веб-сайт – це місце розташування веб-сторінок, які є зв’язаними між собою. До них можна отримати доступ через головну сторінку веб-сайту використовуючи будь-який з доступних веб-браузерів(Internet Explorer, Firefox або Chrome).

1.1. Елементи Html розмітки веб-сторінки

HTML-документ - це текстовий документ, який можна створити, як в звичайному текстовому редакторі, так і використовуючи спеціальні додатки, з підсвічуванням коду.[12]

HTML-документ складається з дерева HTML-елементів і тексту. Кожен елемент позначається в вихідному документі початковим і кінцевим тегом.

DOCTYPE відповідає за коректне відображення веб-сторінки браузером. DOCTYPE визначає не тільки версію HTML (наприклад, html), але і відповідний DTD-файл в Інтернеті.

Елементи, що знаходяться всередині елемента <html>, утворюють дерево документа(рис.1.1), так звану об'єктну модель документа, DOM.

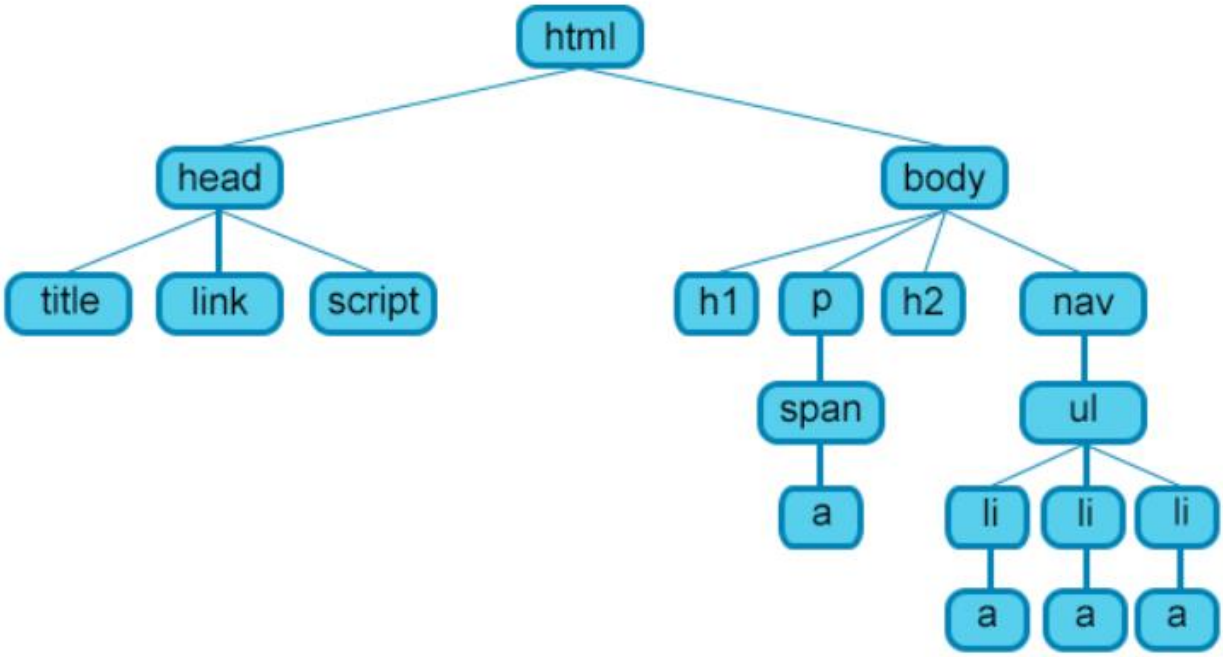


Рис.1.1. «Родинні стосунки» елементів веб-сторінки

Розглянувши взаємодію елементів веб-сторінки, необхідно розглянути так звані «родинні стосунки» між елементами. Відносини між множинними вкладеними елементами поділяються на батьківські, дочірні і сестринські.[13]

- Предок - елемент, який містить в собі інші елементи. На малюнку 1 предком для всіх елементів є <html>. У той же час елемент <body> є предком для всіх розміщених в ньому елементів: <h1>, <p>, , <nav> і т.д.
- Нащадок - елемент, розташований усередині одного або більше типів елементів. Наприклад, <body> є нащадком <html>, а елемент <p> є нащадком одночасно для <body> і <html>.
- Батьківський елемент - елемент, пов'язаний з іншими елементами нижчого рівня, і що знаходиться на дереві вище їх. На малюнку 1 <html> є батьківським тільки для <head> і <body>. Елемент <p> є батьківським тільки для .

- Дочірній елемент - елемент, безпосередньо підлеглий іншому елементу більш високого рівня. На малюнку 1 тільки елементи <h1>, <h2>, <p> і <nav> є дочірніми по відношенню до <body>.
- Сестринський елемент - елемент, який має загальний батьківський елемент з даним, так звані елементи одного рівня. На малюнку 1 <head> і <body> - елементи одного рівня, так само як і елементи <h1>, <h2> і <p> є між собою сестринськими.

1.2. Основні способи стилізації веб-сторінки за допомогою каскадних таблиць стилів

CSS (Cascading Style Sheets) - мова таблиць стилів, що дозволяє прикріплювати стиль до структурованих документів.

Зазвичай CSS-стилі використовуються для створення і зміни стилю елементів веб-сторінок і призначені для користувацьких інтерфейсів, написаних на мовах HTML і XHTML.[13]

Відокремлюючи стиль подання документів від вмісту документів, CSS спрощує створення веб-сторінок і обслуговування сайтів.

Каскадні таблиці стилів описують правила форматування елементів за допомогою властивостей і допустимих значень цих властивостей. Для кожного елемента можна використовувати обмежений набір властивостей, інші властивості не будуть чинити на нього ніякого впливу.

Оголошення стилю складається з двох частин: селектора і оголошення. В HTML імена елементів нечутливі до регістру, тому «h1» працює так само, як і «H1». Селектор повідомляє браузеру, який саме елемент форматувати, а в блоці оголошення перераховуються форматує команди - властивості і їх значення.

Рис. 1.2. Загальний вигляд оголошення стилю

Загальні селектори**:**

- Універсальний селектор(відповідає будь-якому HTML-елементу);
- Селектор елементу(селектори елементів дозволяють форматувати всі елементи даного типу на всіх сторінках сайту) ;
- Селектор класу(селектори класу дозволяють задавати стилі для одного і більше елементів з однаковим ім'ям класу);
- Селектор ідентифікатора(селектор ідентифікатора дозволяє форматувати один конкретний елемент. Значення id має бути унікальним.)
- Селектор нащадка(селектори нащадків застосовують стилі до елементів, розташованим всередині елемента-контейнера.);
- Дочірній селектор(дочірній елемент є прямим нащадком містить його елемента);
- Сестринський селектор(сестринські відносини виникають між елементами, що мають загального батька).

Селектори псевдокласу:

- Link - не відвідуванні посилання;
- Visited - відвідані посилання;

- Hover - будь-який елемент, над яким проводять курсором миші;
- Focus - інтерактивний елемент, до якого перейшли за допомогою клавіатури або активували за допомогою миші;
- Active - елемент, який був активізований користувачем;
- Valid - поля форми, вміст яких пройшло перевірку в браузері на відповідність зазначеного типу даних;
- Invalid - поля форми, вміст яких не відповідає вказаним типом даних;
- Enabled - всі активні поля форм;
- Disabled - заблоковані поля форм, тобто, що знаходяться в неактивному стані;
- In-range - поля форми, значення яких знаходяться в заданому діапазоні;
- Out-of-range - поля форми, значення яких не входять у заданий діапазон;
- Lang () - елементи з текстом на зазначеній мові;
- Not (селектор) - елементи, які не містять вказаний селектор – клас;
- Target - елемент з символом #, на який посилаються в документі;
- Checked - виділені (вибрані користувачем) елементи форми.

Селектори псевдоелементу:

- First-letter - вибирає першу букву кожного абзацу, застосовується тільки до блокових елементів;
- First-line - вибирає перший рядок тексту елемента, застосовується тільки до блокових елементів;
- Before - вставляє генерується вміст перед елементом;
- After - додає генерується вміст після елемента.

1.3. Вибір веб драйверу для проекту

WebDriver - це система автоматизації браузера, яка працює з API з відкритим кодом. Структура функціонує шляхом прийому команд, надсилання цих команд до браузера та взаємодії з програмами. Розглянемо приклади найпоширеніших продуктів, що надають нам його.

1.3.1.Selenium WebDriver

Selenium WebDriver - це програмна бібліотека для управління браузерами. Часто вживається також більш коротка назва WebDriver.[16]

Іноді кажуть, що це «драйвер браузера», але насправді це ціле сімейство драйверів для різних браузерів, а також набір клієнтських бібліотек на різних мовах, що дозволяють працювати з цими драйверами.

Це основний продукт, що розробляється в рамках проекту Selenium.

Selenium WebDriver називається також Selenium 2.0, причина цього буде пояснена нижче.

Як вже було сказано, WebDriver є сімейство драйверів для різних браузерів плюс набір клієнтських бібліотек для цих драйверів на різних мовах програмування:

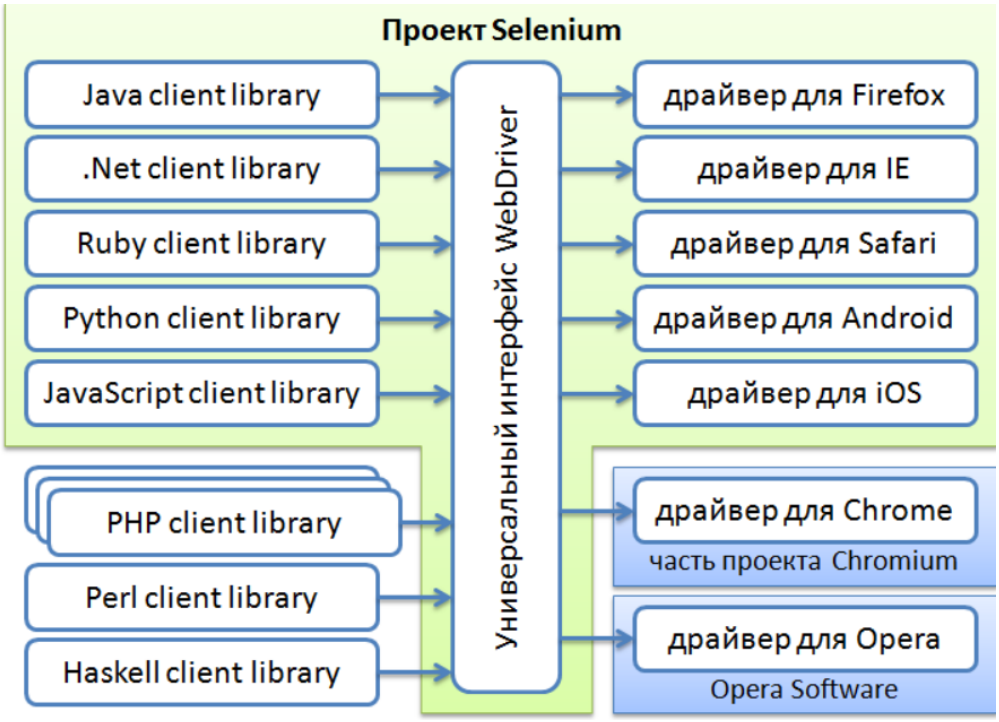


Рис.1.3. Сімейство продуктів Selenium

В рамках проекту Selenium розробляються драйвери для браузерів Firefox, Internet Explorer і Safari, а також драйвери для мобільних браузерів Android і iOS. Драйвер для браузера Google Chrome розробляється в рамках проекту Chromium, а драйвер для браузера Opera (включаючи мобільні версії) розробляється компанією Opera Software. Тому вони формально не є частиною проекту Selenium, поширюються і підтримуються незалежно. Але логічно, звичайно, можна вважати їх частиною сімейства продуктів Selenium.

Аналогічна ситуація і з клієнтськими бібліотеками - в рамках проекту Selenium розробляються бібліотеки для мов Java, .Net (C #), Python, Ruby, JavaScript. Всі інші реалізації не мають відношення до проекту Selenium, хоча, можливо, в майбутньому, якісь з них можуть влитися в цей проект.

Оскільки Webdriver - це інструмент для автоматизації веб додатків, то велика частина роботи з ним це робота з веб елементами (WebElements). WebElements - ні що інше, як DOM об'єкти, що знаходяться на веб сторінці. А для того, щоб

здійснювати якісь дії над DOM об'єктами / веб елементами необхідно їх точним чином визначити (знайти).

1.3.2. TestCafe

TestCafe це інструментальний засіб для автоматизації End-to-End тестування Web додатків заснований на Node.js платформі, який під капотом використовує Proxy.

Що б встановити TestCafe потрібно виконати всього-лише одну команду

npm install -g testcafe

1 хвилина і вуаля, все готово до роботи (привіт, Selenium).

Після цього вам, для тестування вже доступні наступні браузери: Google Chrome, Chromium, Google Chrome Canary, Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Safari.

Також, є вже встановлений Test Runner і ви можете відразу писати і запускати автотести з різними конфігураціями.

Основні компоненти TestCafe можна використовувати з мінімальним знанням мови програмування або навіть без початкових знань, тимчасово, що робить його дуже привабливим для новачків, які можуть з перших днів впроваджувати автоматизацію.

При цьому TestCafe має свій вбудований автоматичний механізм очікування елементів GUI, що в рази полегшує життя автоматизаторів. І звичайно є можливість виставити вручну таймаут очікування, якщо це необхідно.

Ось приклад того, як працює автоматичний механізм очікування для Assertions:

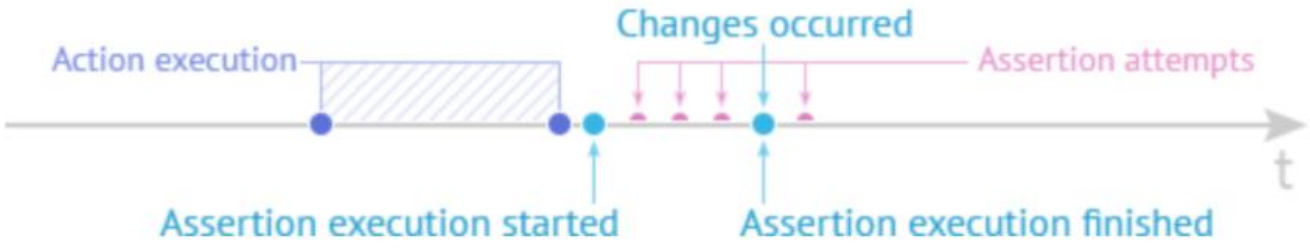


Рис.1.4. Очікування в TestCafe

Після того, як будь-яка дія була виконана, не факт що зміни відбудуться миттєво, це може зайняти якийсь час. Тому що при роботі з Web додатками ми завжди маємо справу з асинхронними операціями. Звідси, перевірка того, чи була очікувана зміна вироблена, відбувається кілька разів. Таким чином ми не отримаємо зафейлений тест, якщо буде певна затримка. Але так само нам не потрібно виставляти вручну довгий таймаут, що в підсумку призведе до збільшення часу прогону тестів.

За допомогою TestCafe тести можна запускати на віддалених пристроях, в тому числі і мобільних. А ключове в цьому те, що на віддаленому пристрої, де будуть ранитися тести, не потрібно інсталювати TestCafe. Для цього потрібно просто перейти по посиланню в браузері на віддаленому пристрої і тести автоматично запустяться. Найголовніше, щоб пристрій на якому встановлений TestCafe і на те, де будуть ранитися тести, були в одній мережі.

Така функціональна можливість буде дуже актуальною, якщо потрібно продемонструвати наявність дефекту на оточенні, де не встановлено TestCafe або навіть сама платформа Node.js. Загалом це дуже зручно.

TestCafe використовує JavaScript, TypeScript і CoffeeScript в своїй роботі, що дозволяє більш гнучко працювати з браузером.

Можливість перехоплювати і мокати запити, паралельно ранити тести, використовувати headless режим, що дуже актуально для зниження часу прогону тестів і інтеграції з CI / CD, гнучко налаштовувати ролі користувачів для тестування

функціональної безпеки, вбудовані hooks, які дозволяють більш гнучко писати тести і багато іншого .

1.3.3.Порівняння двох веб драйверів

Порівняємо TestCafe і Selenium за наступними критеріями:

- Інсталяція та підготовка оточення для автоматизації;
- Простота написання коду і його читабельність;
- Waits в автоматизації;
- Запуск тестів на віддалених пристроях;
- Мова програмування.

Таблиця 1.1.

Порівняння двох веб-драйверів

№ критерія	Selenium	TestCafe
1	-	+
2	+	+
3	+	+
4	-	+
5	+	+

Як бачимо Selenium програє новачку, але код, який ми будемо тестувати є досить давнім, тому краще обрати для нашого проекту ветерана. TestCafe буде корисним для молодих проектів, які тільки розвиваються, а не для мамонтів. Також + Selenium буде те що він дуже гарно працює з мовою програмування Java, яку ми будемо використовувати у планах. TestCafe ж найкраще працює зі своєю рідною

мовою JavaScript. Отже, поміркувавши, було вирішено обрати для проекту Selenium Webdriver.

1.4. Методології розробки програмного забезпечення

1.4.1.Test Driven Development

TDD - це методологія розробки програмного забезпечення, яка ґрунтується на повторенні коротких циклів розробки: спочатку пишеться тест, що покриває бажану зміну, потім пишеться програмний код, який реалізує бажану поведінку системи і дозволить пройти написаний тест. Потім проводиться рефакторинг написаного коду з постійною перевіркою проходження тестів.

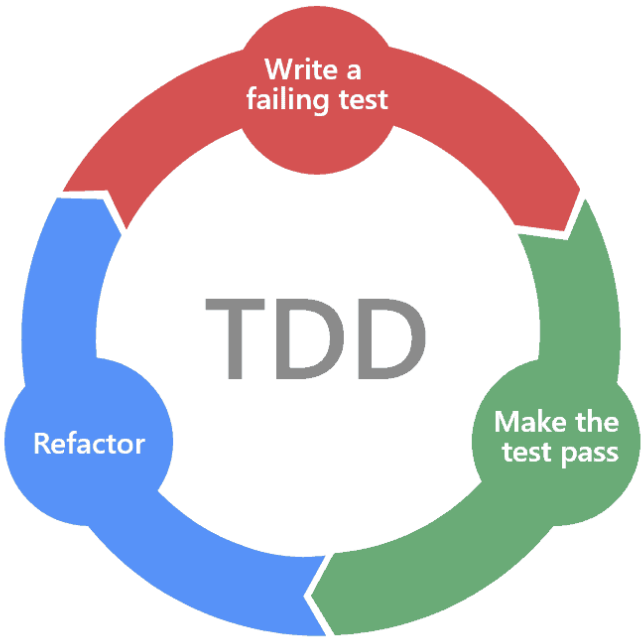


Рис.1.5. Принцип TDD

TDD вважається однією з форм правильного методу побудови програми. Філософія розробки на основі тестів полягає в тому, що ваші тести є специфікацією

того, як ваша програма повинна вести себе. Якщо ви розглядаєте свій набір тестів як обов'язкову частину процесу складання, якщо ваші тести не проходять, програма не збирається, тому що вона неправильна. Звичайно, обмеження полягає в тому, що правильність вашої програми звучить лише як повнота ваших тестів. Проте, дослідження показали, що розробка, заснована на тестуванні, може привести до зниження помилок на 40-80% у виробництві.

Почавши використовувати TDD, ви можете відчути, що працюєте повільніше, ніж зазвичай.

Ця методологія дозволяє домогтися створення придатного для автоматичного тестування програми і дуже гарного покриття коду тестами, так як ТЗ перекладається на мову автоматичних тестів, тобто все, що програма повинна робити, перевіряється. Також TDD часто спрощує програмну реалізацію: виключається надмірність реалізації - якщо компонент проходить тест, то він вважається готовим.

Архітектура програмних продуктів, що розробляються таким чином, зазвичай краще. Стабільність роботи програми, розробленої через тестування, вище за рахунок того, що всі основні функціональні можливості програми покриті тестами і їх працездатність постійно перевіряється. Супроводжуваність проектів, де тестується все або практично все, дуже висока - розробники можуть не боятися вносити зміни в код, якщо щось піде не так, то про це повідомлять результати автоматичного тестування.

1.4.2. Type Driven Development

Type Driven Development скорочено пишеться також, як і розробка через тестування, тому зазвичай пишуть повну назву.

Рис.1.6. Type Driven Development «на пальцях»

При розробці на основі типів ваші типи даних і сигнатури типів є специфікацією програми. Типи також служать формою документації, яка гарантовано оновлюється.

Типи являють собою невеликі контрольні точки, завдяки яким, ми отримуємо безліч міні-тестів по всьому нашому додатку. Причому витрати на створення типів мінімальні і актуалізувати їх не потрібно, так як вони є частиною кодової бази.

Розробка за типом - це ще один правильний метод побудови програми. Як і в разі розробки на основі тестування, розробка на основі типів може підвищити вашу впевненість в коді і заощадити ваш час при внесенні змін у велику кодову базу.

З мінусів тільки зростаюча складність у мов з динамічною типізацією. Наприклад, для JavaScript цей підхід важче застосувати, ніж для TypeScript.

1.4.3. Behavior Driven Development

Через деякі методологічного подібності TDD (Test Driven Development) і BDD (Behaviour Driven Development) часто плутають навіть професіонали. У чому ж відмінність? Концепції обох підходів схожі, спочатку йдуть тести і тільки потім починається розробка, але призначення у них абсолютно різний. TDD - це більше про програмування і тестування на рівні технічної реалізації продукту, коли тести створюють самі розробники. BDD передбачає опис тестувальником або аналітиком користувальницьких сценаріїв на природній мові - якщо можна так висловитися, на мові бізнесу.

BDD - behaviour-driven development - це розробка, заснована на описі поведінки. Певна людина (або люди) пише опису виду "я як користувач хочу коли натиснули кнопку пуск тоді показувалося меню як на картинці". Програмісти давно написали спеціальні Тули, які подібні описи переводять в тести. А далі класична розробка з тестами.[17]

Якщо записувати назви тестів у вигляді пропозицій і при запису імен методів використовувати лексику бізнес-домену, створена документація стає зрозуміла замовникам, аналітикам і тестувальникам.

- Тексти сценаріїв записуються в певній формі;
- Маючи якийсь контекст;
- Коли відбувається подія;
- Тоді перевірити результат.

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario: Add two numbers
    Given I have entered 50 into the calculator
      And I have entered 70 into the calculator
     When I press add
     Then The result should be 120 on the scree
```

Рис.1.7 Приклад сценарію згідно з BDD

BDD підхід спільно з інженерними практиками дозволив нам відмовитися від legacy-документації, що містить неактуальну інформацію, і отримувати нову документацію на льоту, зберігати її разом з проектом, що наблизило аналітиків і тестувальників до коду.

BDD - скоріше, процес, метою якого є здешевлення реалізації нових фіч. Ще на старті розробки ми отримуємо важливі артефакти. Наприклад, зрозумілу для підтримки документацію. Ця документація дає можливість усім зацікавленим особам сформувати своє уявлення про продукт і сценаріях користувача поведінки, які повинні бути реалізовані в ході ітерацій розробки. З BDD-підходом ми також знижуємо поріг входу в проект нових учасників.

Плюси:

- Тести читаються не програмістами;
- Їх легко змінювати. Вони часто пишуться майже чистою англійською;
- Їх може писати product owner або інші зацікавлені особи;
- Результати виконання тестів більш "людяні";
- Тести не залежать від цільової мови програмування. Міграція на іншу мову сильно спрощується.

Але у даного підходу є і недоліки - це довго і дорого. BDD незручний хоча б тим, що вимагає залучення фахівців тестування вже на етапі опрацювання вимог, а це подовжує цикл розробки.

Виходом з цієї ситуації може виявитися вибір відповідного BDD фреймворка і правильно збудованих процесів розробки.

Багато вже давно зрозуміли, що тестування - це свого роду панацея від усіх хвороб, але чи так це насправді? Безумовно, грунтовно протестований код працює стабільніше і передбачувані, але тести не позбавляють нас від проблем і помилок на етапі проектування і постановки завдань. Наступні підходи до розробки можуть допомогти вам з цим.

1.4.4. Feature Driven Development

FDD - Ця методологія (коротко іменована FDD) була розроблена Джеффом Де Люка (Jeff De Luca) і визнаним гуру в області об'єктно-орієнтованих технологій Пітером Коад (Peter Coad). FDD являє собою спробу об'єднати найбільш визнані в індустрії розробки програмного забезпечення методики, які беруть за основу важливу для замовника функціональність (властивості) розроблюваного програмного забезпечення. Основною метою даної методології є розробка реального, працюючого програмного забезпечення систематично, в поставлені терміни.

Рис.1.8. Етапи Feature Driven Development

Як і інші адаптивні методології, вона робить основний упор на коротких ітераціях, кожна з яких служить для опрацювання певної частини функціональності системи. Згідно FDD, одна ітерація триває два тижні. FDD налічує п'ять процесів. Перші три з них відносяться до початку проекту:

- розробка загальної моделі;
- складання списку необхідних властивостей системи;
- планування роботи над кожним властивістю;
- проектування кожного властивості;
- конструювання кожного властивості.

Останні два кроки необхідно робити під час кожної ітерації. При цьому кожен процес розбивається на завдання і має критерії верифікації.

Давайте детальніше зупинимося на кожному пункті.

Розробка загальної моделі.

Розробка починається с аналізу широти наявного кола завдань і контексту системи. Далі для кожної моделюється області робиться більш детальний розбір. Попередні опису складаються невеликими групами і виносяться на подальше обговорення і експертну оцінку. Після одна із запропонованих моделей або їх сукупність стає моделлю для конкретної області. Моделі кожної області завдань

об'єднуються в загальну підсумкову модель, яка може змінюватися протягом роботи.

Складання списку функцій

Інформація, зібрана при побудові загальної моделі, використовується для складання списку функцій. Функції об'єднуються в так звані "області" (англ. Domain), а вони ж у свою чергу діляться на під області (англ. Subject areas) за функціональною ознакою.

Кожна під область відповідає певному бізнес-процесу, а його кроки стають списком функцій (властивостей). Функції представлені у вигляді «дія - результат - об'єкт», наприклад, «перевірка пароля користувача». Розробка кожної функції повинна займати не більше 2 тижнів, інакше завдання необхідно деком позувати на більш дрібними ітерації. Список властивостей в FDD - те ж саме, що і product backlog в SCRUM.

План за властивостями (функцій)

Далі йде етап розподілу функцій серед провідних програмістів або по командам.

Проектування функцій

Для кожного властивості створюється проектувальний пакет. Ведущий програміст виділяє невелику групу властивостей для розробки протягом двох тижнів. Після залишаються докладні діаграми послідовності для кожного властивості, уточнюючи загальну модель. Далі пишуться «заглушки» класів і методів. У цей момент ми повинні сфокусуватися на дизайні програмного продукту.

Реалізація функції

Пишемо код, прибираємо заглушки, тестуємо.

Після того, як властивість протестовано і пішло в продукт, беремо наступне за пріоритетами властивість, повторюємо цикл дизайну / реалізації.

Разом, в результаті ми отримуємо:

- документація по властивостям системи;
- ретельне проектування;
- простіше оцінювати невеликі завдання;
- тести орієнтовані на бізнес-завдання;
- опрацьований процес створення продукту;
- короткі ітеративні цикли розробки дозволяють швидше нарощувати функціональність і зменшити кількість помилок.

Мінуси:

- FDD більше підходить для великих проектів. Невеликі команди розробки не зможуть відчути всі переваги даного підходу;
- значні витрати на впровадження і навчання.

1.4.5. Panic Driven Development

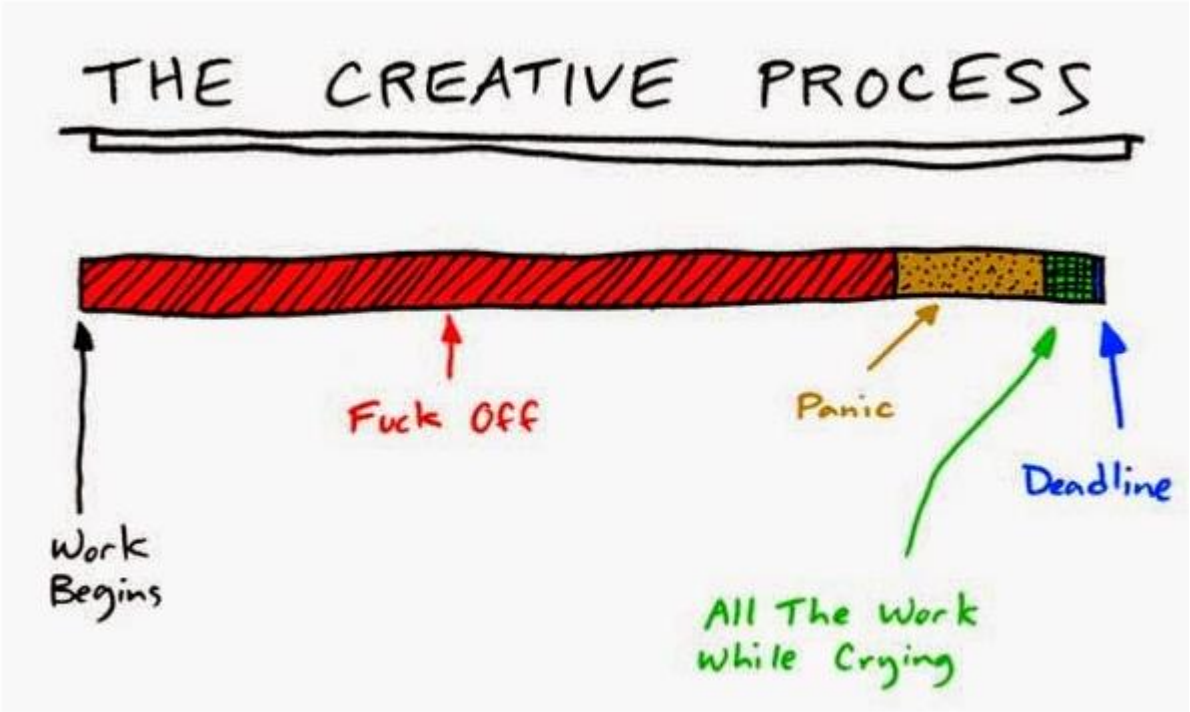


Рис.1.9. Процес PDD

Нові завдання пріоритетнішими за старі.

Всякий раз, коли в середині спринту з'являється нова проблема, вона має пріоритет над будь запланованої роботою. Нове завжди краще і має більш високий пріоритет. Дивно, чому це не стало одним із принципів гнучкої розробки? Націленість на забезпечення цінності для клієнта вимагає, щоб команда дбала про нові фічах і відкладала раніше певну роботу.

Пишіть стільки коду, скільки потрібно, щоб вирішити проблему.

Розробники пишуть код для життя. Помилки можуть бути виправлені лише кодом. Обговорення дизайну та UX може тільки сповільнити розробку. Але ми ж не хочемо втрачати дорогоцінний час? Спочатку напишіть рішення, потім перевірте своє припущення щодо виправлення. Якщо виправлення працює, проблема вирішена.

Тести повинні писатися в кінці.

Після того, як виправлення впроваджено, тести можуть бути заплановані як завдання, яке буде зроблена в майбутньому. Тести корисні, але не є пріоритетними. Ви можете подбати про них пізніше. Ручного тестування повинно бути достатньо, щоб довести працездатність реалізованого рішення.

Довіритеся своєму інстинкту.

Програмування - це мистецтво. Мистецтво має внутрішню інстинктивну складову. Довірся своїй інтуїції. Напишіть код. Розгорніть його. Тільки сміливим посміхається удача.[18]

Процес гнучкий.

Будь-який процес, створений для розробки, тестування та випуску програмного забезпечення, - це просто набір угод і правил, які не висічені в камені. Критичні виправлення вимагають різних підходів. Очікується, що ви зігнеться процес, щоб виконати завдання в строк, якщо цього вимагає бізнес.

Це процес, керований менеджером.

Як частина однієї команди, менеджери мають право висловити свою думку з питань розвитку. Рефакторинг або передовий досвід можуть і повинні бути скасовані потребами бізнесу. Інженери можуть висловити свою думку, але вони повинні в кінцевому підсумку прийняти будь-які потреби, які приходять зверху.

Плюси підходу:

- висока швидкість розробки;
- дешево;
- замовники щасливі, що нарешті-то знайшли тямущих розробників.

мінуси:

- всі плюси розіб'ються об технічний борг і складність проекту.

PDD своєрідний антипаттерн розробки, який, на жаль, ми весь час від часу практикуємо.

1.4.6. Порівняння усіх методологій

Опрацювавши увесь вище описаний матеріал, було обрано технологію BDD, через високу поширеність та зрозумілість «пересічній» людині. Весь проект буде писатися використовуючи цю технологію. Щодо інших технологій вони є або занадто складними, або занадто простими для даного проекту.

1.5. Середовища розробки програмного забезпечення

1.5.1. IntelliJ IDEA

IntelliJ IDEA - інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains.

Перша версія з'явилася в січні 2001 року і швидко набула популярності як перше середовище для Java з широким набором інтегрованих інструментів для рефакторингу, які дозволяли програмістам швидко реорганізовувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.[19]

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного інтерфейсу користувача. Серед інших можливостей, середа добре сумісна з багатьма популярними вільними інструментами розробників, такими як CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування на мові Ruby .

Починаючи з версії 9.0, середа доступна в двох редакціях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією, доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Kotlin, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В редакції Ultimate Edition, доступною під комерційною ліцензією, реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов та фреймворків.

Створення нового проекту в IntelliJ IDEA. Програма пропонує вибір мов

програмування.

Мови:

- Javal;
- JavaScript;
- CoffeeScript;
- HTML / XHTML / HAML;
- CSS / SASS / LESS;
- XML / XSL / XPath;
- YAML;
- ActionScript / MXML;
- Python;
- Ruby;
- Haxe;
- Groovy;
- Scala;
- SQL;
- PHP;
- Kotlin;
- Clojure;
- Ci;
- C ++;
- Go;
- Rust (за допомогою офіційного плагіна).

Ряд мов підтримується за допомогою плагінів сторонніх розробників, зокрема, так реалізована підтримка OCaml, GLSL, Erlang, Fantom, Haskell, Lua, Mathematica, Perl5, Object Pascal.

1.5.2. Eclipse

Eclipse є безкоштовною програмною платформою з відкритим вихідним кодом, контролюється організацією Eclipse Foundation. Написана на мові програмування Java і основною метою її створення є підвищення продуктивності процесу розробки програмного забезпечення.[20]

Претендує на статус найбільш популярною Java IDE і є єдиним конкурентом такої потужної платформи як NetBeans.

Але на відміну від NetBeans який для створення елементів призначеного для користувача інтерфейсу використовує від платформи незалежну бібліотеку Swing, в Eclipse використовується платформо-залежна бібліотека SWT - Standard Widget Toolkit.

IDE розроблені на базі платформи Eclipse застосовуються для створення програмного забезпечення на різних мовах програмування, так як Eclipse є платформою для розробки будь-яких інтегрованих середовищ програмування і розширень для себе ж, за принципом "Додатки для Eclipse розробляються в самій Eclipse".

Особливості платформи Eclipse:

- Кросплатформеність - працює під операційними системами Windows, Linux, Solaris і Mac OS X;
- Використовуючи Eclipse можна програмувати на багатьох мовах, таких як Java, C і C ++, PHP, Perl, Python, Cobol та інших;
- Є фреймворком для розробки інших інструментів і пропонує великий набір API для створення модулів;
- Використовуючи підхід RCP (Rich Client Platform) Eclipse є інструментом для створення практично будь-якого клієнтського програмного забезпечення.

Робота над проектом Eclipse ведеться в декількох напрямках, основні три - робота над платформою Eclipse, розробка Java IDE, розробка плагінів для розширення функціональності Eclipse.

Гнучкість і розширюваність досягається завдяки модульності платформи.

Архітектура платформи Eclipse

Основним елементом є виконуюча п'ятниця - Eclipse Runtime, в якій виконуються коди розширень і модулів. Вона забезпечує всю базову функціональність платформи - управління розширеннями і оновленнями, взаємодія з операційною системою, забезпечення роботи системи допомоги.

Наступним елементом є власне IDE - вона відповідає за управління основними елементами програми, їх розташуванням і настройками, управління проектами, налагодження та складання проектів, пошук по файлах і командну розробку.

У стандартну поставку Eclipse SDK включені два плагіна - Java Development Tools або JDT, і Plug-in Developer Environment або PDE, таким чином ми отримуємо повністю готову IDE для Java програмування і для розробки розширень для Eclipse.

Eclipse SDK - це мінімальна версія, ідеальна для знайомства з платформою і навчання. Надалі, визначившись з цілями, ви можете завантажити і використовувати будь-яку відповідну для ваших завдань складання, вже укомплектовану необхідними розширеннями.

Приклади інших спеціалізованих збірок Eclipse

- Eclipse IDE for Java Developers - середовище розробки на мові Java;
- Eclipse IDE for Java EE Developers - середовище розробки веб додатків і корпоративних додатків з використанням технології Java EE;
- Eclipse IDE for C / C ++ Developers - функціональна IDE для програмування на C і C ++;
- Eclipse IDE for JavaScript Web Developers - IDE для розробки веб додатків з використанням HTML, XML, JavaScript і CSS.

Можна завантажити вже готову збірку, необхідну вам для роботи або попрацювати і встановити необхідні модулі. В рамках даної статті звернемо увагу на базовий пакет - Eclipse SDK.

Його можна завантажити на сайті http://www.eclipse.org/downloads, в процесі завантаження вам буде запропоновано вибрати зі списку свою операційну систему, потім ви будете перенаправлені на сторінку з вибором територіально близького дзеркала для скачування.

Вибирайте Eclipse Classic - цей пакет повністю готовий для розробки додатків на Java і плагінів для Eclipse.

1.5.3. Net beams

NetBeans IDE - безкоштовна інтегрована середовище розробки з відкритим вихідним кодом для розробників програмного забезпечення. Середовище надає всі засоби, необхідні для створення професійних десктоп додатків, корпоративних, мобільних і веб-додатків на платформі Java, а також C / C ++, PHP, JavaScript, Groovy і Ruby.[21]

Офіційний сайт: http://netbeans.org/index_ru.html

Основні характеристики NetBeans IDE:

- Робоча область середовища IDE є повністю настроюється - існує можливість для користувача настройки дій, які виконуються за допомогою панелі, призначення "гарячих" клавіш і т.д;
- IDE має в своєму складі розширений багатомовний редактор для різних мов програмування - Java, C / C ++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, RHTML, JSP, документацію Javadoc. Існує можливість розширення функцій редактора з метою підтримки будь-якого іншого мови;

- Редактор NetBeans робить відступи рядків, перевіряє відповідність дужок і слів, підсвічує синтаксис вихідного коду;
- Проводиться перевірка помилок під час введення, відображення варіантів для автозавершення коду і фрагментів документації по вашій мові програмування;
- Редактор може генерувати і вставляти в вихідний код стандартні фрагменти коду на Java або іншими мовами;
- Браузер класів дозволяє переглядати ієрархію і структуру будь-якого класу Java - відображаються інтерфейси, базові класи, похідні класи і члени класів;
- Існує можливість переміщення будь-якої вкладки редактора в межах робочого простору IDE і за її межі, створюючи незалежну вікно, яке можна перемістити на другий екран;
- Можливість групування пов'язаних проектів - створюючи групи проектів, можна швидко відкривати і закривати кілька згрупованих проектів одночасно;
- Розширені засоби для виконання контекстно-залежного пошуку по всій середовищі IDE, довідкових матеріалів і всіх відкритих проектів і файлів;
- Існує можливість створення проектів у вільному форматі або починати роботу з проектом з шаблону. У комплекті з середовищем IDE поставляються шаблони і приклади проектів для додатків Java SE, мобільних, веб-додатків і додатків рівня підприємства, додатків JavaFX, модулів NetBeans, додатків Groovy, PHP, C / C ++, Ruby і Ruby on Rails;
- NetBeans IDE є платформою для побудови десктоп додатків з функціональним призначеним для користувача інтерфейсом, тому що вдає із себе фреймворк до Java бібліотеці Swing;
- NetBeans має вбудовану підтримку CVS, Mercurial і Subversion. Для

перегляду змін використовується редактор з кольоровими позначками.

Можливості програмування в NetBeans:

- Розробка Java десктоп додатків з професійними графічними інтерфейсами користувача. Використовується візуальний редактор - Swing GUI Builder. Робота здійснюється шляхом перетягування елементів графічного інтерфейсу з палітри на полотно. Попереднє позиціонування елементів можна здійснювати за допомогою покажчика миші. Панель властивостей і інспектор компонентів надають можливість тонкої настройки кожного компонента інтерфейсу;
- Створення веб-додатків і корпоративних додатків відповідно до стандартів. Середа NetBeans надає повну підтримку Java EE 6. Дозволяє розробляти веб-сторінки, сервлети, веб-сервіси, Enterprise Java Beans (EJB), проекти Java EE з використанням JavaServer Faces 2.0 (Facelets), Spring, Struts і Hibernate;
- Програмування на PHP, підтримка всіх супутніх мов програмування, технологій і веб-стандартів. Можливість створювати проекти PHP на основі платформи Zend або Symfony. Редактор PHP динамічно інтегрований з функціями редагування HTML, JavaScript і CSS. Проекти PHP можуть бути розгорнуті з середовища NetBeans на локальному або віддаленому сервері при взаємодії через FTP або SFTP;
- Можливість створення, тестування, налагодження і впровадження програм, що функціонують на мобільних телефонах, кишенькових комп'ютерах, телеприставки і вбудованих системах. Visual Mobile Designer (VMD) створює всю необхідну модульну інфраструктуру проекту та забезпечує швидку розробку графічних інтерфейсів шляхом перетягування в робочу область компонентів - екран очікування, екран входу в систему, оглядач файлів, засіб складання повідомлень SMS і екран заставки.

Можливість створення призначеного для користувача інтерфейсу на основі SVG;

- Використання JavaFX Composer для візуального структурування додатки JavaFX з графічним інтерфейсом, аналогічно конструктору GUI Swing для Java десктоп додатків;
- Можливість розробки професійних програм на мовах C, C ++ для різних платформ - Windows, Linux, Mac і Solaris. Підтримуються всі широко використовувані компілятори - GNU, Cygwin і MinGW. Існує можливість установки необхідного компілятора, визначень препроцесора, параметрів часу компіляції і т.д;
- Розширені можливості по роботі з базами даних - вбудований клієнт до баз даних - MySQL, Postgres, Oracle та ін., Редактор запитів SQL, можливість редагувати таблиці баз даних безпосередньо через редактор таблиць;
- Інтеграція з серверами додатків і контейнерами сервлетів - автоматичне розгортання додатків, управління сервером - запуск, зупинка, перезапуск;
- Багатомовний користувальницький інтерфейс з підтримкою російської мови;
- Розширення функціональності за допомогою модулів, гнучка система управління компонентами, модулями, оновленням та завантаженням модулів через інтернет.

NetBeans - єдина IDE, яка влаштує і початківця розробника і професіонала. Наявність докладної вбудованої довідкової системи забезпечить швидкий старт для початківців.

1.5.4. Порівняння усіх середовищ розробки

З усього вище сказаного, було вирішено обрати середовищем розробки IntelliJ IDEA, оскільки дане середовище розробки є найбільш універсальним(за рахунок великої кількості мов та плагінів) та має найзручніший інтерфейс користувача.

Також для досягнення нашої мети буде цілком достатньо безкоштовної версії даного середовища. Також великою перевагою IntelliJ є досвід роботи автора даного проекту з ним та наявність уже готового налаштованого середовища, тобто не потрібно буде тратити час на інсталяцію та налаштування продукту та звикати до нового графічного інтерфейсу середовища. Інші середовища були відкинуті через брак бажаного функціоналу чи незрозумілість інтерфейсу користувача.

Висновки до розділу 1

В ході виконання робіт пов’язаних з цим розділом було досліджено та проаналізовано основні складові веб-сторінки та веб-сайту. Це робилося з метою визначення того, як має виглядати веб-сторінка, що на ній може бути та як це можна протестувати. У процесі було визначено основні складові веб-сторінки(веб-сайту) та способи ними маніпулювати або змінювати їх стан чи вигляд.

Було розглянуто основні складові html-документа та їх вплив на відображення сторінки браузером. Також було розглянуто методи доступу до самих елементів документу та способи зміни їх станів.

Також було проведено дослідження оформлення html-документу за допомогою каскадних таблиць стилів. Дослідили основні способи взаємодії та вибірки елементів по селекторах що буде дуже важливо у подальшій роботі над 2 розділом.

Розглянуто основні найпоширеніші методології тестування та обрану найбільш підходящу для даного проекту. Також було обрано найзручніше середовище

розробки ПЗ.

В результаті отримано структуру базових елементів WEB - сторінки пов’язаних з тим, як протестувати структуру на коректність та правильність без розгортання самого проекту веб-сторінки локально, а з допомогою веб-браузера та спеціальних інструментів

РОЗДІЛ 2
РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ТЕСТУВАННЯ ВЕБ-СТОРІНОК

2.1. Формулювання завдання

Для розробки програмного модуля тестування наповненості контентом веб-сторінок та їх базового функціоналу, на базі якого можна було б створювати тести різноманітних властивостей веб-сайту та забезпечити якісне тестування необхідно:

1) Сформулювати вимоги до програмного модуля, що розробляється;
2) Побудувати архітектуру програмного модуля;
3) Вибрати необхідні компоненти для комплексного рішення вимог;
4) Розробити проект;
5) Вбудувати обрані компоненти для функціонування програмного модуля;
6) Налагодити програмний модуль;

2.2. Формулювання вимог до програмного модуля

Існує багато факторів, які можуть мати вплив на кінцеву якість тестового програмного модуля. Для створення якомога надійнішого рішення щодо автоматизації тестування наповненості контентом веб-сторінок та їх базового функціоналу, необхідно врахувати якість модуля тестування, адже він є специфічним програмним забезпеченням, яке використовуються для діагностики основних складових та функціоналу веб-сторінки. Отже, програмний продукт має відповідати наступним критеріям якості:

- Надійність: Тестовий фреймворк повинний бути значно надійнішим, ніж тестоване програмне забезпечення. Рівень його надійності має максимально можливим при наявних обставинах та ресурсах;
- Практичність використання: Модуль повиннен бути простим у використанні. Уся інформація вихідна інформація чи інтерфейс користувача має бути зрозумілим пересічному користувачеві;
- Переносимість: Модуль повиннен працювати на операційних системах типу Windows, Mac, Linux та хмарних сховищах. Якщо цю умову не виконати, то корисність та сфера застосування програмного застосунку значно зменшується;
- Звітність: звіти мають містити повну інформацію про тестовий випадок чи помилку, викладену у лаконічній формі та читабельній формі;
- Інтеграція: Фреймворк має вбудовуватися в інші системи легко та швидко.

Для повної функціональності тестовий програмний продукт повинен мати наступні функції:

1) Ітерації наборів даних. Дана функціональність дасть можливість

виконувати одні і ті ж набори тестових кроків, але при запускаючи їх з різними вхідними тестовими даними. Прикладом можна привести тестовий сценарій авторизації, коли при одній і тій же послідовності кроків вводяться різні логіни і паролі, що дає змогу перевірити декілька тестових сценаріїв не пишучи окремі тести для кожного з них.

2) Звіти результатів тестування. Для даного програмного продукту необхідно створити можливість генерування звітних даних по певних періодах чи окремих сценаріях чи взагалі по окремих тестах.

3) Логування дій. Для кращого розуміння що саме могло піти не так і де саме, важливим є додавання функції логування кроків тесту. Це дасть можливість швидше знайти помилку та виправити її без значної шкоди та з найменшими можливими витратами часу.

4) Використання готових, сучасних бібліотек для UI тестування, які наддадуть можливість спростити сам процес тестування та зробити його більш зрозумілим.

2.3. Опис архітектури програмного модуля тестування

Архітектура або багатошарова архітектура часто використовується, як основа для архітектури автоматизації тестування. Цей шаблон архітектури відповідає традиційним IT-комунікаційним та організаційним структурам, які знаходяться в більшості світових компаній, що робить його природнім та очікуваним вибором для більшості програмних рішень.

Компоненти в багаторівневої архітектури організовані в горизонтальні рівні, кожен з яких виконує певну призначену тільки йому роль у програмі. Хоча багатошаровий шаблон архітектури не визначає точно кількість та тип шарів, які мають бути в шаблоні, більшість багаторівневих архітектур складаються з 4

стандартних шарів: рівень представлення, рівень додатку, бізнес рівень та рівень доступу до даних. Деколи бізнес рівень та рівень додатку об'єднуються в єдиний рівень[28]. Одним з найбільших плюсів багаторівневої структури архітектури є розділення проблем між компонентами. Компоненти певного рівня мають справу лише з логікою, що стосується цього рівня. Дана класифікація компонентів дозволяє легко будувати ефективні моделі ролей та відповідальності у архітектурі, а також дозволяє легко розробляти, тестувати, керувати та підтримувати програми за допомогою цього шаблону архітектури завдяки чітко визначеним інтерфейсам компонентів та обмеженій області використання рівнів.

Концепція ізоляції рівнів означає, що зміни, які були внесені в один з рівнів архітектури не впливають на компоненти інших рівнів. Зміна відбувається тільки на цьому рівні, та, можливо на рівні пов’язаному з ним.

Дана архітектура як найкраще підходить для побудови програмного модуля тестування наповненості контентом веб-сторінок та їх базового функціоналу, оскільки дозволяє виконувати поставлені вимоги до нашої програми. Завдяки використання багаторівневого шаблону можна розділити логіку тестів, процеси обробки даних та представлення звітів. Це рішення дає змогу використовувати різноманітні бібліотеки, фреймворки та існуючі компоненти.

Програмний модуль, що розроблюється, складається з чотирьох рівнів архітектури:

- рівень доступу до даних;
- тестовий рівень;
- основний рівень (додатку);
- рівень представлення.

Рівень доступу до даних відповідає за тестові дані, необхідні для створення різноманітних тестів. В програмі необхідно реалізувати зберігання та передачу статичних і динамічних даних.

Тестовий рівень необхідний для відокремлення логіки тестів від їх реалізацій. Завдяки цьому наявна можливість створити параметризовані, атомарні та незалежні тести. Завдяки цьому, тести можна об’єднувати в групи, розподіляти за рівнями та типами.

За безпосереднє виконання самих тестів та їх обробку відповідає рівень додатку. Саме він є ядром модулю, яке виконує основну роботу. Основний рівень поєднує в собі механізми створення запитів до веб-сторінки, обробку відповідей від неї та перевірку.

Рівень представлення несе відповідальність за створення та відтворення звітності та зберігання логованої інформації.

2.4. Опис програмного модуля тестування веб-сторінок та його компонентів

Згідно з поставленими до програмного модуля завданнями та встановленими вимогами, необхідно розробити фреймворк з відкритим вихідним кодом для написання тестів, які направлені на тестування наповненості контентом веб-сторінок та їх базового функціоналу. Мета створення даного фреймворку полегшення написання тестів та генерація звітів про результати тестування.

Основною метою програми, що розробляється, виконання таких функцій:

- створення запуск готових тест-кейсів або властивостей;
- створення нових тестових сценаріїв;
- об’єднання тестових сценаріїв у групи(властивості)
- представлення тестових сценаріїв допомогою технології BDD та скриптової мови Gerkin;
- генерація звітів і виконаних тестів.

Програмний модуль має підтримувати інтеграцію з іншими фреймворками і інтеграція в систему CI/CD для виконання безперервного тестування та звітності.

В результаті вибору програмних продуктів і технологій обрано мову Java 1.8 в інтегрованому середовищі розробки IntelliJ IDEA Community Edition та використати наступні технології:

1) Java;
2) JUnit;
3) BDD(Cucumber);
4) Selenium WebDriver.

2.5. Розроблення базової структури сторінок та формування PFM

Найперше нам потрібно буде знайти наші Web-Element на сторінках нашого сайту та прописати взаємодію з ними. Це ми зробимо за допомогою локаторів які будемо шукати вручну безпосередньо на веб-сторінці(рис.2.1) та заносячи ці локатори у певні відображення наших сторінок. На цих же відображеннях сторінок ми будемо прописувати методи взаємодії з елементами, для кращої ізоляції та дотримання PFM(рис.2.2). Давайте приступимо.

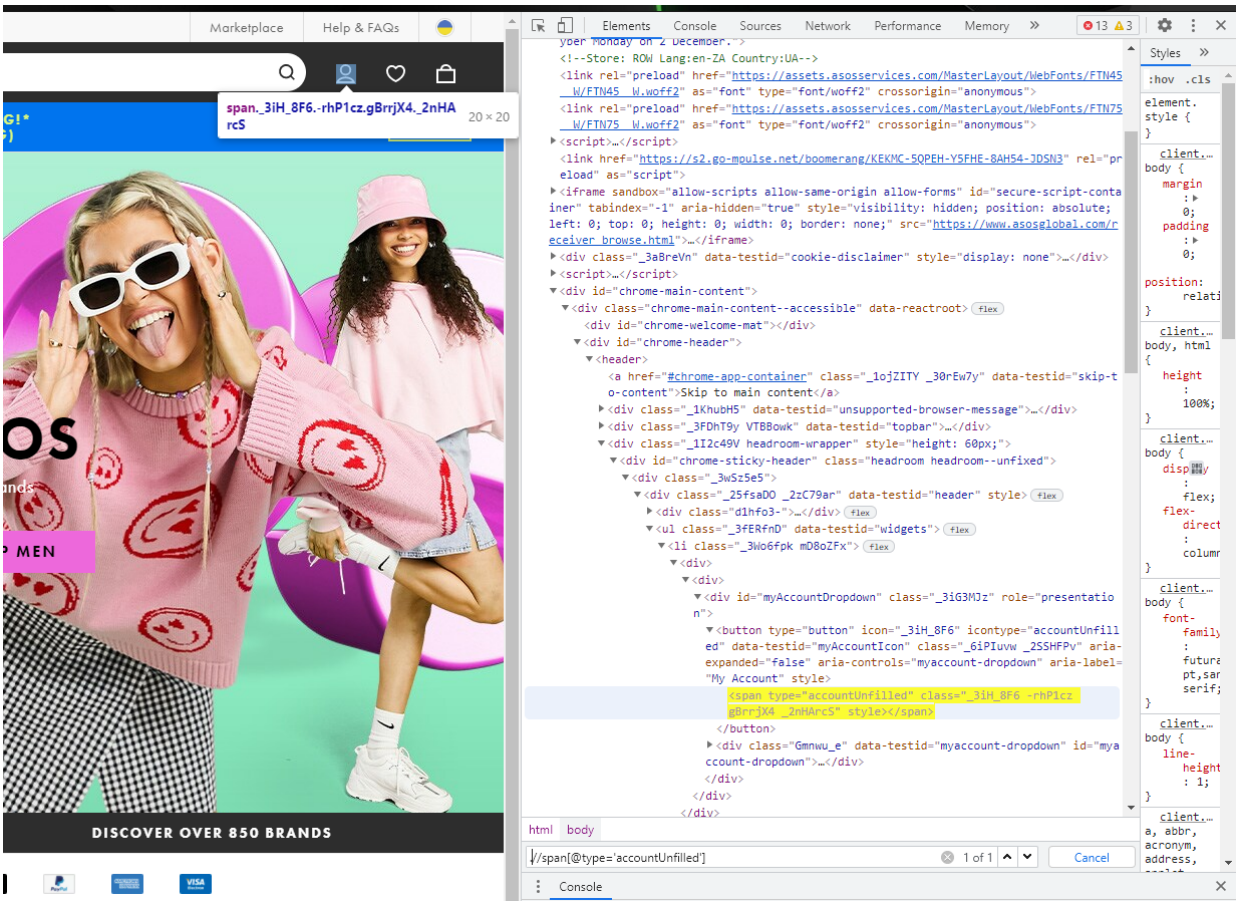


Рис.2.1. Приклад пошуку локатора

```java
public class HomePage extends BasePage {

    private static final String SEARCH_WORD1="skirt";
    private static final String SEARCH_WORD2="t-shirt";
    private static final String SEARCH_WORD_MEN="Gym Clothes";
    public HomePage(WebDriver driver) { super(driver); }
    @FindBy(xpath = "//span[@type='accountUnfilled']")
    private WebElement loginIcon;
    @FindBy(xpath = "//a[@data-testid='myaccount-link']")
    private WebElement loginLink;

    @FindBy(xpath = "//a[@data-testid='women-floor']")
    private WebElement womenWear;

    @FindBy(css = "input#chrome-search")
    private WebElement searchField;

    @FindBy(xpath = "//a[@data-testid='men-floor']")
    private WebElement menWear;

    @FindBy(xpath = "//img[@alt='nike']")
    private WebElement nikeButton;

    public WebElement returnLoginLink() { return loginLink; }
    public void clickOnLoginIcon() { loginIcon.click(); }

    public WebElement getLoginLink() { return loginLink; }

    public void clickOnLogin() { loginLink.click(); }
```

Рис.2.2. Приклад класу Сторінки з локаторами та методами

2.6. Формування BDD

Саме BDD ми можемо формувати 2 способами – спочатку створити логіку тестів у файлах .feature(рис.2.3) за допомогою «зрозумілої» мови Gerkin, а потім уже крокам цих тестів підв’язати реалізацію(рис.2.4) за допомогою PFM, або ж піти з

точністю та навпаки – спочатку зробити реалізацію, а потім оформлення. Я ж обрав перший спосіб, оскільки він є більш зрозумілим та швидким, як на мене.

Отже розпочнемо.

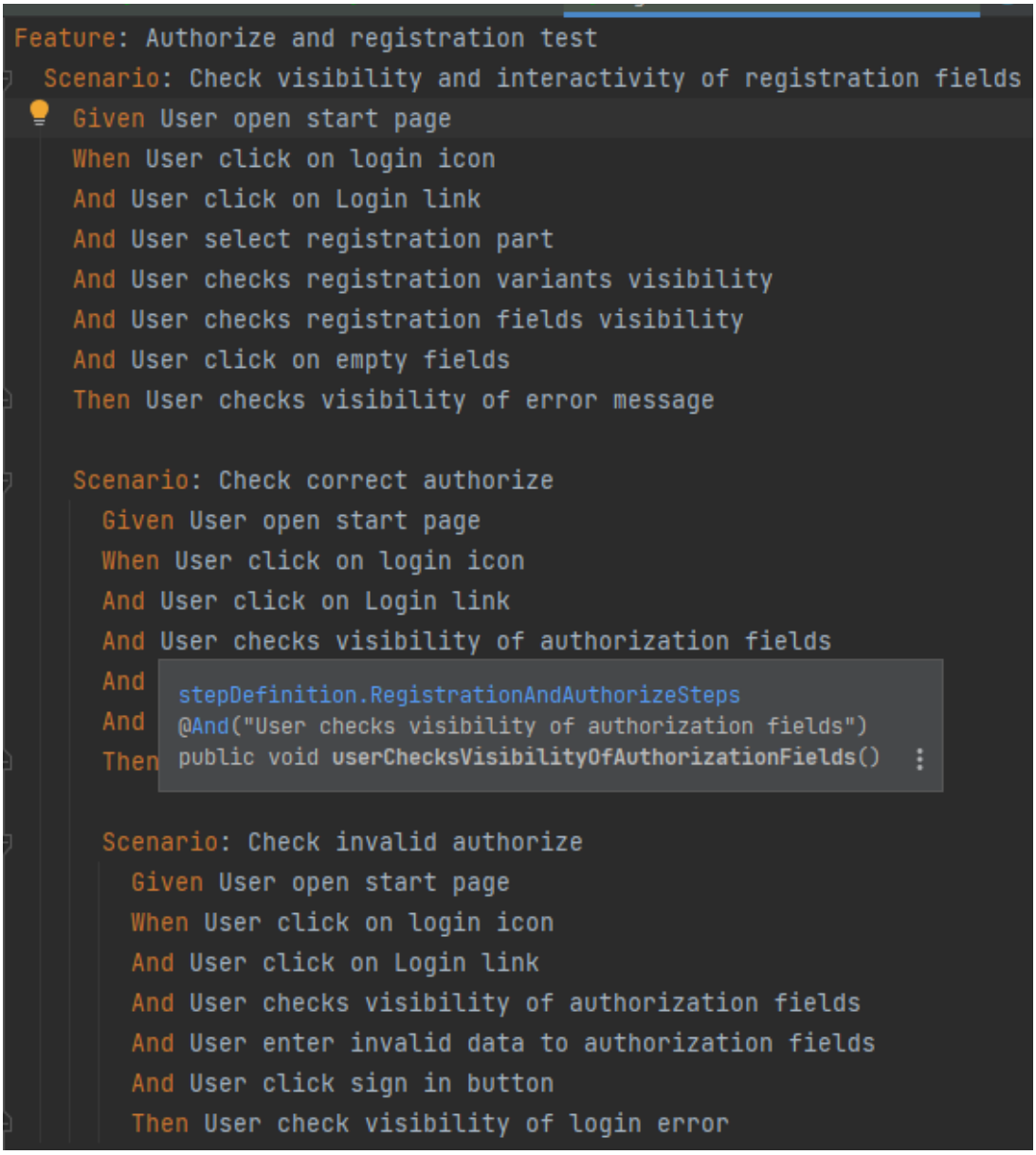


Рис.2.3. Приклад файлу .feature

```java
public class RegistrationAndAuthorizeSteps {
    private static final String ASOS_URL = "https://www.asos.com/";
    private static final long DEFAULT_TIMEOUT = 30;
    WebDriver driver;
    HomePage homePage;
    AuthorizePage authorizePage;
    RegestrationPage regestrationPage;
    PageFactoryManager pageFactoryManager;

    @Before
    public void testsSetUp() {
        chromedriver().setup();
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        pageFactoryManager = new PageFactoryManager(driver);
    }

    @Given("User open start page")
    public void userOpenStartPage() {
        homePage = pageFactoryManager.getHomePage();
        homePage.openHomePage(ASOS_URL);
    }

    @When("User click on login icon")
    public void userClickOnLoginIcon() { homePage.clickOnLoginIcon(); }

    @And("User click on Login link")
    public void userClickOnLoginLink() {
        homePage.waitVisibilityOfElement(DEFAULT_TIMEOUT, homePage.returnLoginLink());
        homePage.clickOnLogin();
    }

    @And("User select registration part")
    public void userSelectRegistrationPart() {
        AuthorizePage authorizePage = pageFactoryManager.getAuthorizePage();
        authorizePage.clickOnReg();
    }
```

Рис.2.4. Приклад реалізації кроків тесту

2.7. Налагодження тестів та перевірка «вчасності» всіх дій

Тепер коли всі тести сформовано їх обов’язково потрібно прогнати декілька разів(краще декілька десятків) з різним ступенем навантаження на систему. Потрібно перевірити чи ніде не зламався локатор і чи вистачає часу очікування елементу. Якщо ж тест зафейлиться, то потрібно буде пройти його у режимі

дебагу(рис.2.5)(покрокового викання), та перевірити чи фейлиться він і так. Якщо ні то вся проблема у недостатньому часі очікування і додатковий вейтер є рішенням. Коли всі тести будуть налагоджені, продукт можна вважати готовим до експлуатації.

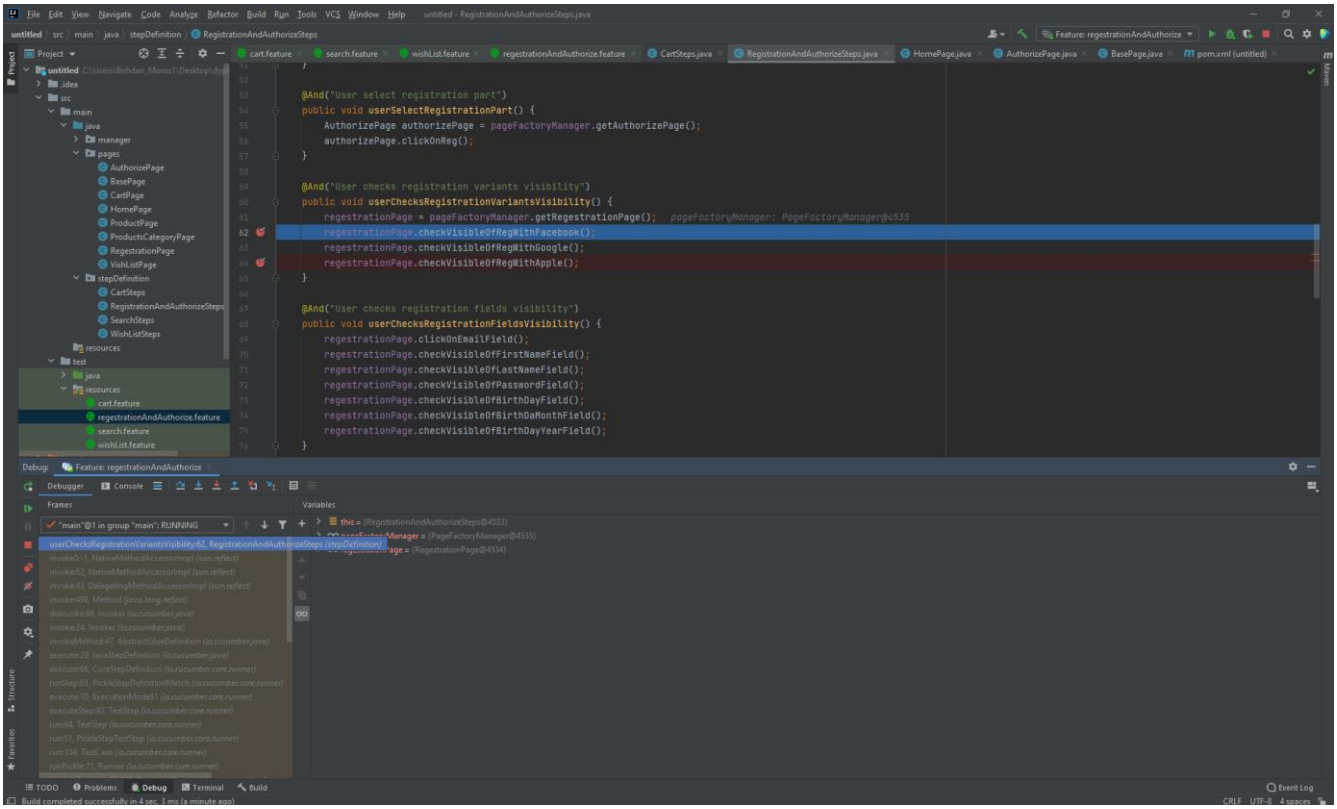


Рис.2.5. Приклад роботи з відладкою

Тепер наш програмний продукт повністю готовий до введення в експлуатацію. Можна трішки відсвяткувати 😊.

2.8. Висновки до розділу 2

У другому розділі встановлено вимоги та завдання до програмного модуля. В результаті чого розроблена багаторівнева архітектура, на основі якій спроектована

та розроблена інфраструктура програмних рішень на мові програмування Java. Архітектура програмного модуля поєднує в собі чотири рівня: доступ до даних, тестовий, основний та відображення. Для реалізації архітектури інтегровані та налаштовані наступні технології: Maven, Junit, Selenium, Cucumber, PFM.

Було визначено локатори усіх потрібних елементів та записано їх у змінні на відповідних сторінках-класах. Також було сформовано PFM за допомогою менеджера сторінок та розподілу усього функціоналу за сторінками. Пізніше було створено тестові сценарії за допомогою мови Gerkin та записано реалізацію кроків цих сценаріїв у спеціальні класи. Потім це все було відполіровано багаторазовим запуском тестових сценаріїв з різним ступенем завантаженості системи та на різних тестових середовищах(веб-браузерах).

В підсумку отримано програмне рішення для тестування базового функціоналу та наповненості контентом веб-сторінок. На базі програмного модуля можна проводити тестування найпоширеніших вразливостей веб-сторінок, створювати тести та групувати в тестові набори, проводити аналіз відповідей сервера, генерувати звіти тестування в комфортній формі.

Оскільки програмний модуль представлений у вигляді фреймворка з відкритим вихідним кодом, його можна інтегрувати з іншими фреймворками тестування.

РОЗДІЛ 3
ПІДТРИМКА У ВИКОРИСТАННІ ПРОДУКТУ

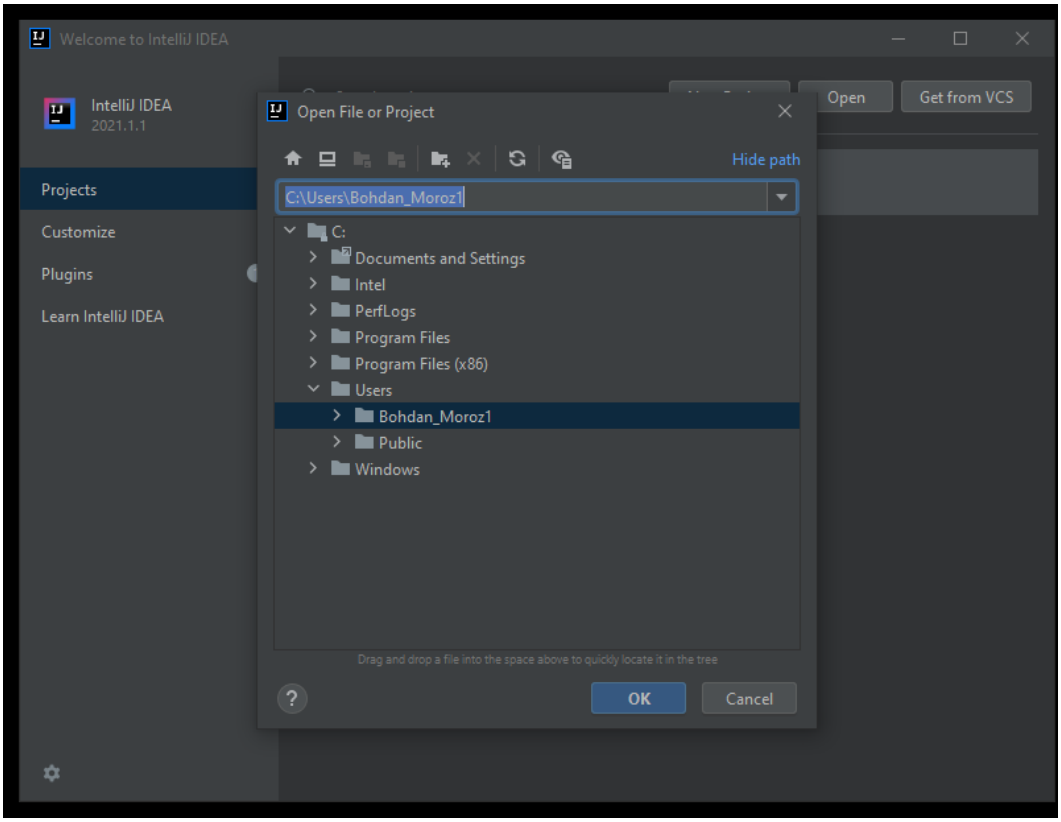
3.1. Інструкція з використання даного фреймворку

Даний програмний продукт можна використовувати двома способами – перший спосіб є більш зручним та зрозумілішим, однак вимагає додаткового програмного забезпечення. Другий же є візуально складнішим, однак не потребує додаткових налаштувань. Розглянемо обидва способи.

У будь якому випадку вам потрібно інсталювати SDK\JDK вище 8 версії та Maven на свій пристрій. Після цього потрібно завантажити сам проект з гугл сховища. Посилання можна попросити у автора.

3.1.1. Зрозуміліший спосіб

Для цього вам знадобиться програмний продукт під назвою IntelliJ Idea. Інсталювавши його вам просто достатньо відкрити у ньому проект(рис.3.1) вибравши його розташування у контекстному меню та запустити бажані тестові сценарії(рис.3.3) у відкритому проекті(рис.3.2).



Рисю3.1. Вибір папки з продуктом

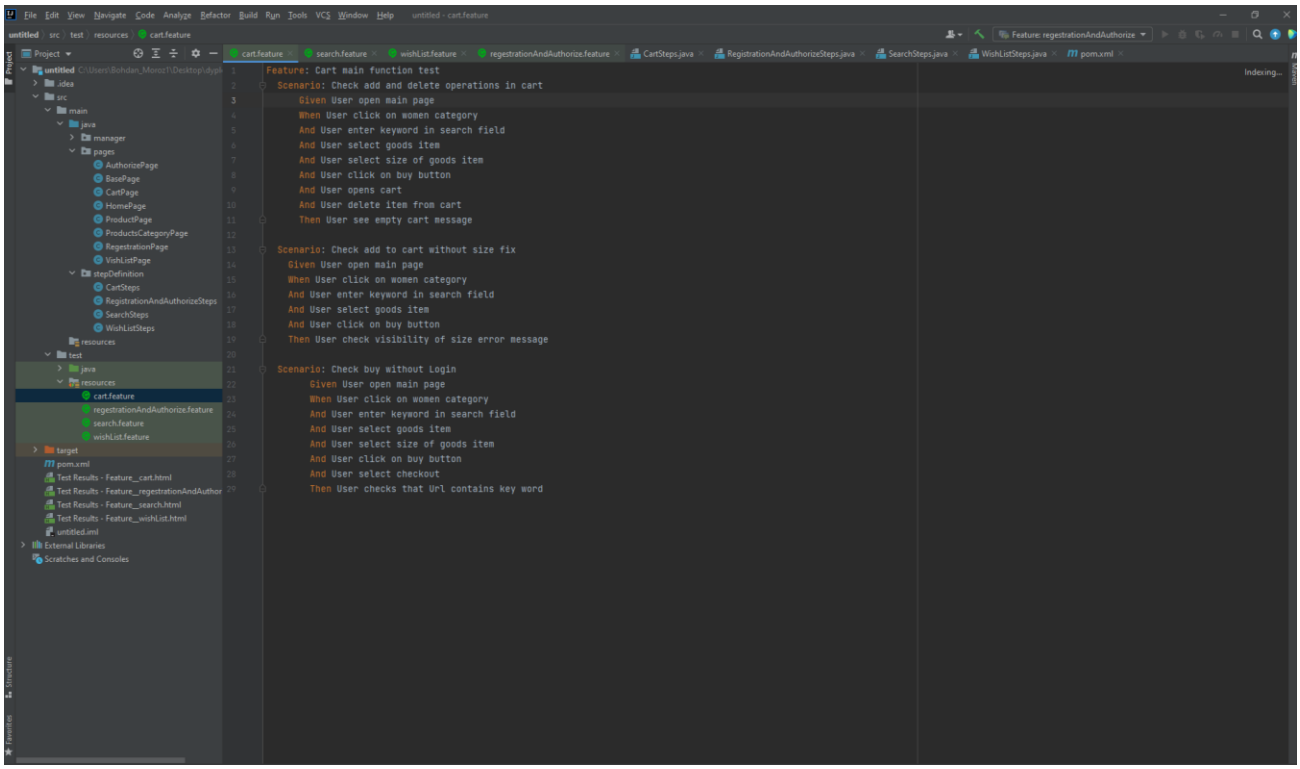


Рис.3.2. Відкритий проект з файлом .feature

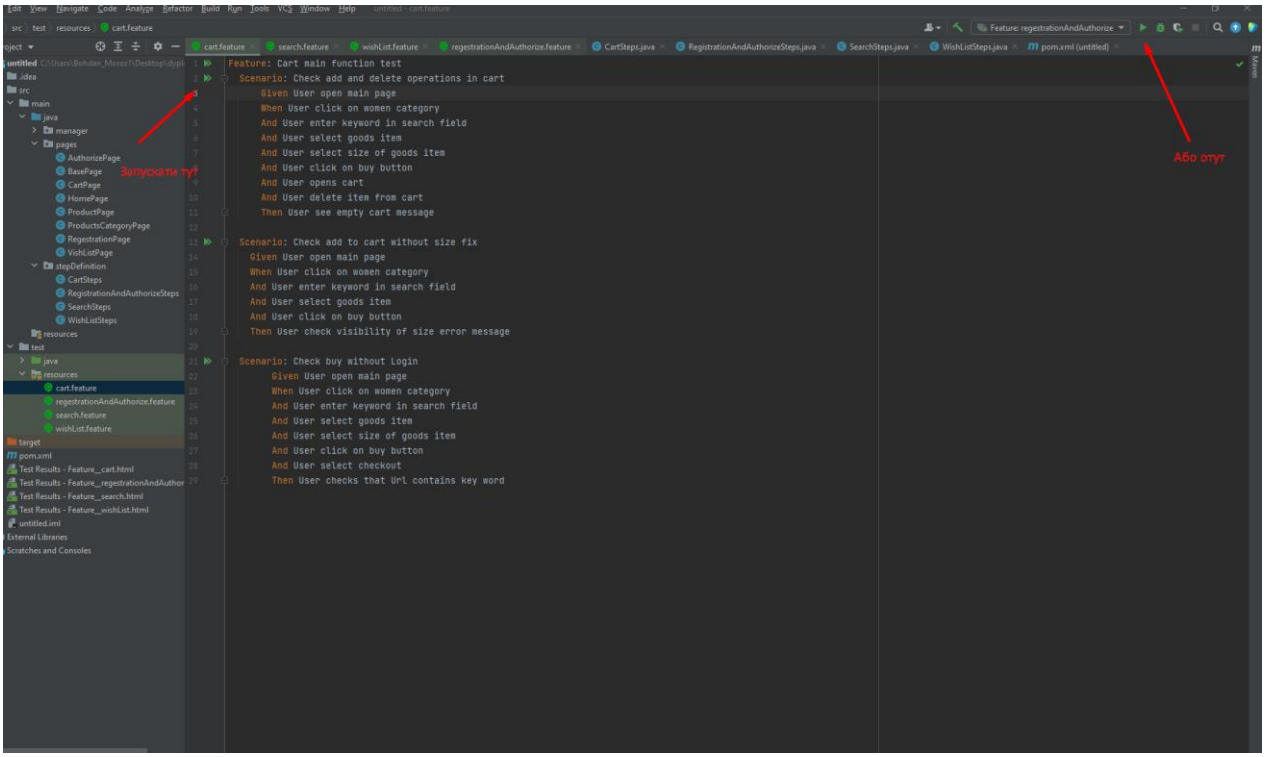


Рис.3.3. Інструкція з запуску

Після того як ви запустите весь набір тестів чи окремий тест відкриється браузер(рис.3.4) та почнуть виконуватися кроки тесту. По завершенню виконання вам буде повідомлено про вдачу чи провал(рис.3.5).

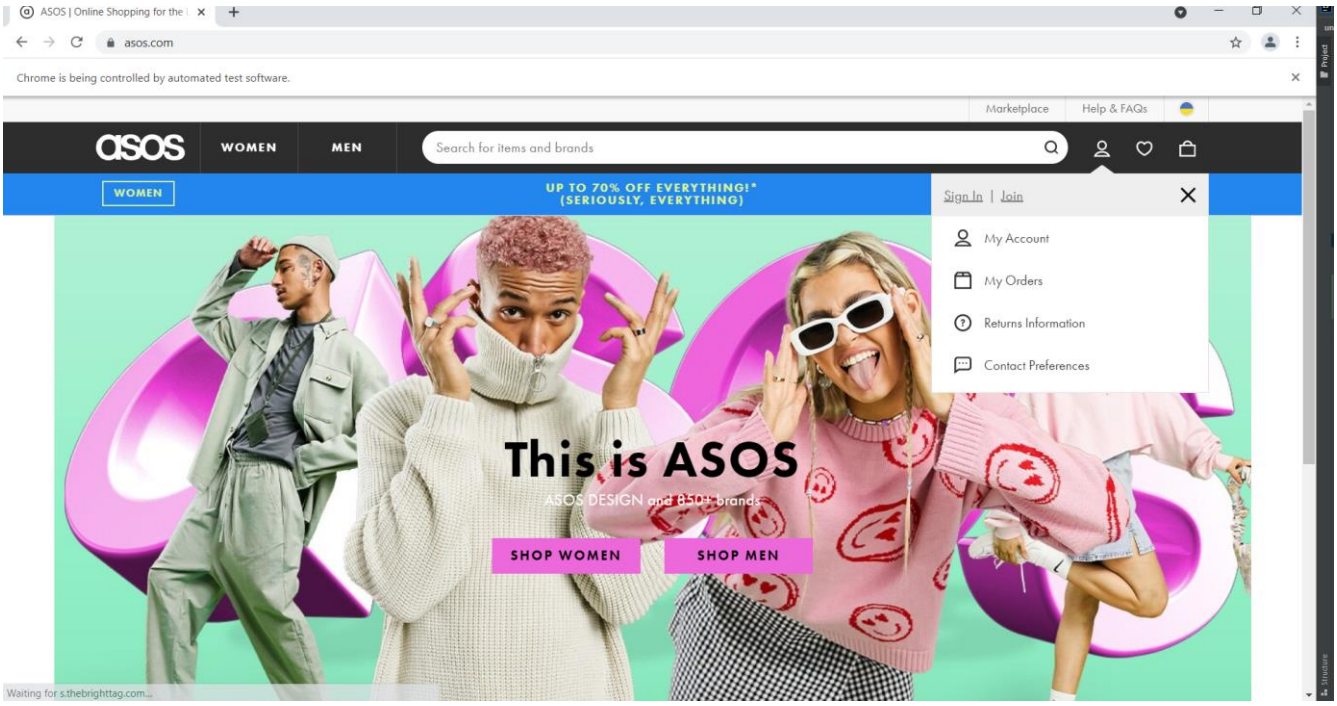


Рис.3.4. Вікно браузера з виконанням тесту

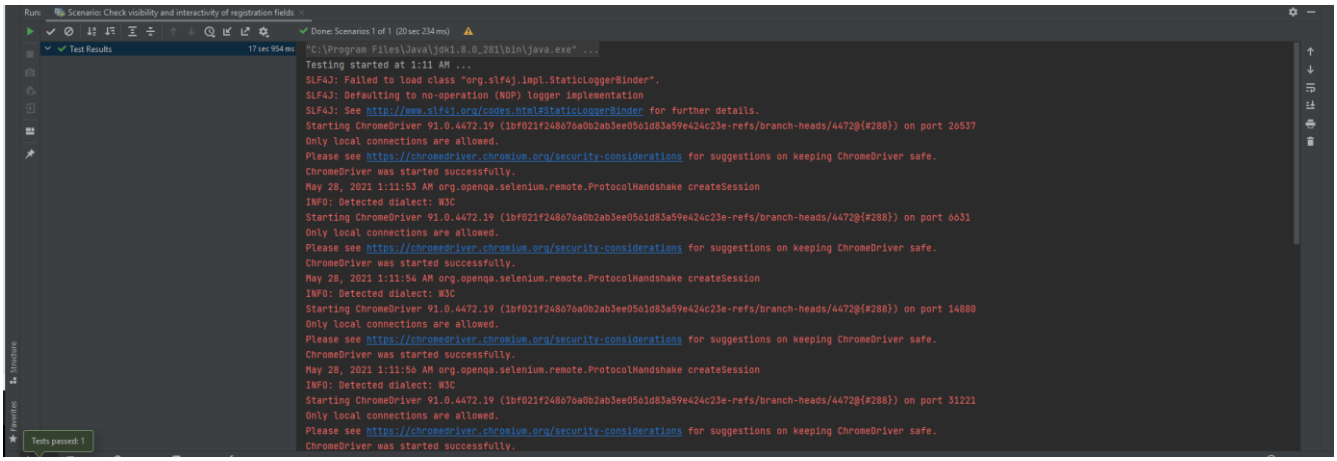


Рис.3.5. Повідомлення про успішність тесту

Потім ви можете сформувати більш зрозумілий покроковий звіт натиснувши спеціальну кнопку(рис.3.6), вибравши формат даних(рис.3.7) та відкривши його у браузері(рис.3.8).

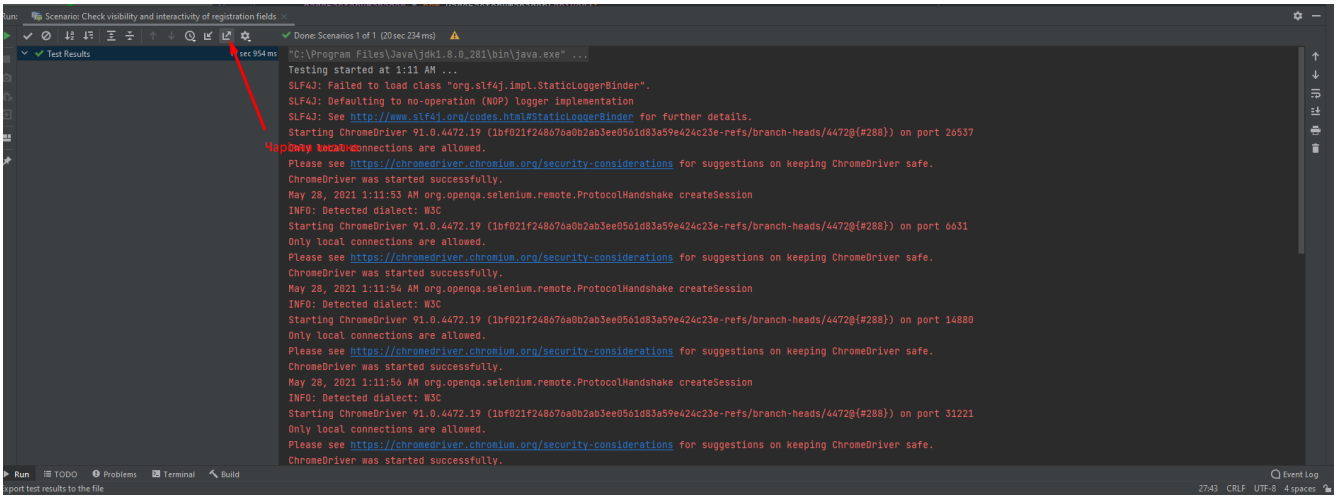


Рис. 3.7. Магічна кнопка

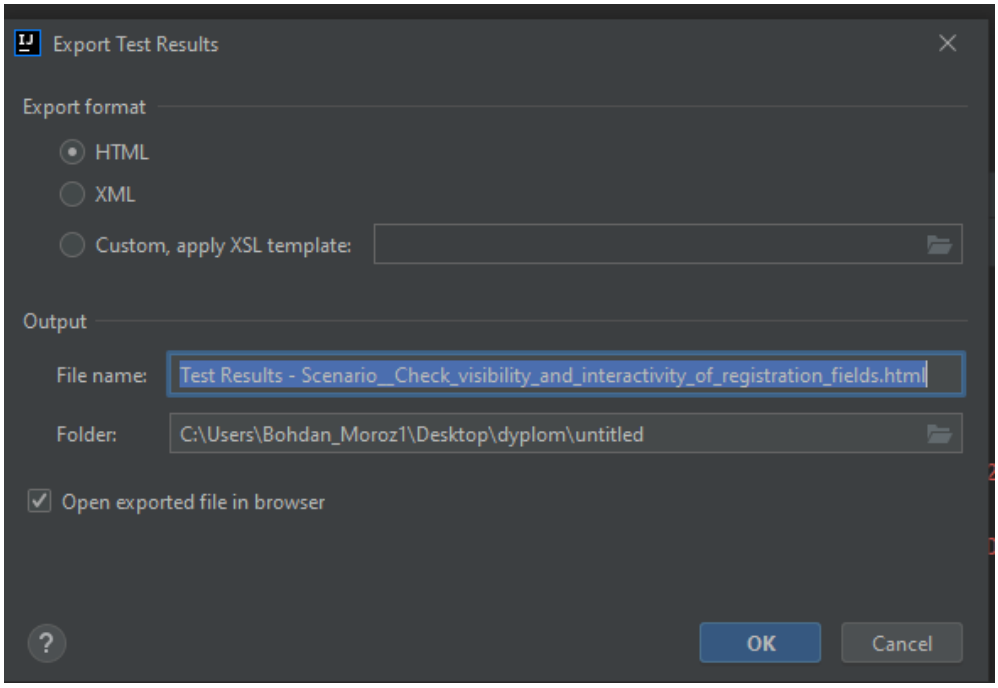


Рис.3.7. Вибір формату даних

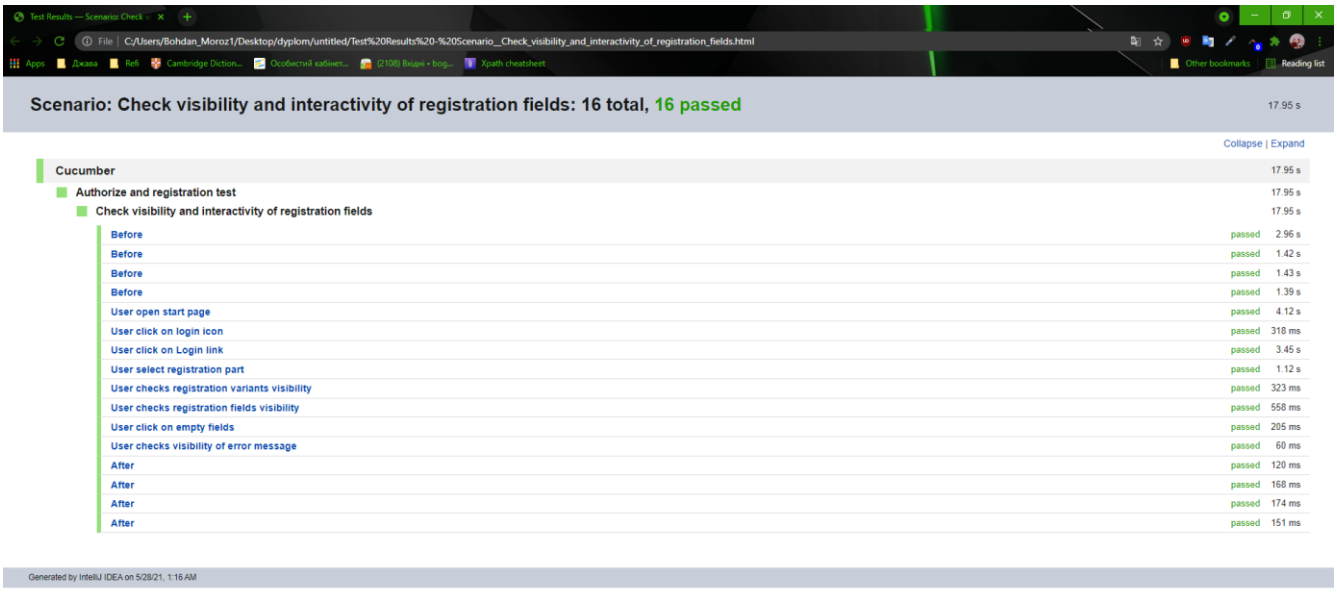


Рис.3.8. Вигляд звіту в браузері

На цьому перший варіант можна рахувати завершеним

3.1.2. Текстовий варіант

Даний спосіб потребує тільки браузер, завантажений проект та вміння працювати з командним рядком. Отже спершу вам потрібно відкрий командний рядок та перейти у директорію з проектом. Потім якщо ви плануєте використовувати Junit потрібно виконати таку команду:

java -cp <classpath> org.junit.runner.JUnitCore com.example.test.RunCukesTest

,де RunCukesTest – це юніт тест що встановлює параметри Cucumber.

Також можна використовувати cucumber-jvm в командному рядку:

```
java -cp <classpath> cucumber.api.cli.Main \
   --glue com.example.test \
   --plugin pretty path/to/features
```

Однак я б не рекомендував цей спосіб через незручний інтерфейс та можливість легко допустити помилку.

ВИСНОВОК

У дипломному проекті було досліджено базовий функціонал та внутрішню будову веб-сторінок та розроблено програмний модуль для підвищення ефективності тестування базового функціоналу та наповненості контентом веб-сторінок.

Для досягнення поставленої цілі було вивчено функціонування веб-сторінок, проаналізовано найпоширеніші підходи до побудови веб-сторінок та сучасний стан існуючих програмних рішень для тестування базового функціоналу та наповненості контентом веб-сторінок. .

Веб-сторінки поєднують в собі масу різних технологій та способів реалізацій, що в свою чергу, породжує велику кількість помилок та багів. Після аналізу функціонування веб-сторінок та їх внутрішньої будови, виникає розуміння необхідності проведення тестування базового функціоналу та наповненості контентом. Оскільки на ринку не існує універсального рішення для комплексного тестування, було запропоновано спроектувати та розробити новий, унікальний програмний модуль, де буде можливо інтегрувати різні технології та налаштовувати інфраструктуру під вимоги проекту. Для вирішення цієї задачі були встановлені вимоги до створеного програмного рішення. Була розроблена багаторівнева архітектура, на основі якій було спроектовано та розроблено програмне рішення на мові програмування Java. Програмний модуль, завдяки поєднанню сучасних технологій, може проводити тестування найпоширеніших вразливостей веб-сторінки. Програмний додаток поєднує в собі такі функції, як: зберігання та передача тестових даних, створення та перевірка тестових сценаріїв, виконання різноманітних методів HTTP, подача створених тестів зрозумілою для пересічної людини(З рівнем Англійської А2) манерою, тестування в різних

поширених браузерах та платформах, формування детальних звітів про проведене тестування. Всі ці функції реалізовані за допомогою інтеграції компонентів Maven, Junit,Selenium, Cucumber, PFM. Оскільки проект виконаний на мові java, він є кроссплатформеним та підтримується на різних операційних системах. Також створений програмний модуль легко інтегрується в системи безперервного тестування.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2019 Trustwave Global Security Report [Електронний ресурс] // Trustwave. – 2019. – Режим доступу: https://www.trustwave.com/en-us/resources/library/documents/2019-trustwave-global-security-report/. (дата звернення 12.05.2021р). – Назва з екрана.

2. Zero-day (computing) [Електронний ресурс] // wikipedia. – 2018. – Режим доступу: https://en.wikipedia.org/wiki/Zero-day_(computing). (дата звернення 12.05.2021р). – Назва з екрана.

3. Лекция 2: Введение в клиент-серверные технологии Веб. Протокол HTTP [Електронний ресурс] // ИНТУИТ. – 2015. – Режим доступу: https://www.intuit.ru/studies/courses/485/341/lecture/8182?page=1. (дата звернення 13.05.2021р). – Назва з екрана.

4. HTTP [Електронний ресурс] // MDN web docs. – 2018. – Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTTP. (дата звернення 13.05.2021р). – Назва з екрана.

5. Міжнародний стандарт ISO/IEC 29147 [Електронний ресурс]. – Режим доступу:http://standards.iso.org/ittf/PubliclyAvailableStandards/c045170_ISO_IEC_29147_2014.zip#en. (дата звернення 13.05.2021р). – Назва з екрана.

6. Міжнародний стандарт ISO/IEC 27000 [Електронний ресурс]. – Режим доступу:http://standards.iso.org/ittf/PubliclyAvailableStandards/c066435_ISO_IEC_27000_2016(E).zip. (дата звернення 13.05.2021р). – Назва з екрана.

7. База уязвимостей Secunia [Електронний ресурс] // FlexEra. – 2016. – Режим доступу : https://www.flexera.com/products/operations/software-vulnerability-research/secunia-research/advisories.html. (дата звернення 15.05.2021р). – Назва з екрана.

8. Руководство по Maven. Введение. [Електронний ресурс] // PROSELYTE –

Режим доступу : https://proselyte.net/tutorials/maven/introduction/. (дата звернення 15.05.2021р). – Назва з екрана.

9. Краткое знакомство с Maven [Електронний ресурс] // Tproger. – 2019. – Режим доступу : https://tproger.ru/articles/maven-short-intro/. (дата звернення 15.05.2021р). – Назва з екрана.

10. JUnit – Краткое руководство [Електронний ресурс] // Coderlessons. – 2019. – Режим доступу : https://coderlessons.com/tutorials/java-tekhnologii/vyuchit-junit/junit-kratkoe-rukovodstvo. (дата звернення 15.05.2021р). – Назва з екрана.

11. Assertion in JUnit [Електронний ресурс] // splessons – Режим доступу : https://www.splessons.com/lesson/assertion-in-junit/. (дата звернення 15.05.2021р). – Назва з екрана.

12. HTML 5 [Електронний ресурс] – Режим доступу: http://htmlbook.ru/html5 (дата звернення 15.05.2021р). – Назва з екрана.

13. Справочник по CSS [Електронний ресурс] – Режим доступу : http://htmlbook.ru/css (дата звернення 16.05.2021р). – Назва з екрана.

14. Сучасний підручник JavaScript [Електронний ресурс] – Режим доступу: https://learn.javascript.ru/ (дата звернення 16.05.2021р). – Назва з екрана.

15. HTML & CSS: Design and Build Web Sites Дж. Дакет [Електронний ресурс] // – 2011 - Режим доступу: https://wtf.tw/ref/duckett.pdf (дата звернення 16.05.2021р). – Назва з екрана.

16. Selenium WebDriver Practical Guide, C. Авасарала [Електронний ресурс] // – 2014, - Режим доступу: http://padabum.com/d.php?id=173763 (дата звернення 16.05.2021р). – Назва з екрана.

17. Руководство по Cucumber [Електронний ресурс] – Режим доступу: https://proselyte.net/tutorials/cucumber/ (дата звернення 17.05.2021р). – Назва з екрана.

18. Патерни проектування [Електронний ресурс] – Режим доступу:

https://refactoring.guru/uk/design-patterns (дата звернення 17.05.2021р). – Назва з екрана.

19. IntelliJ Idea як середовище розробки [Електронний ресурс] – Режим доступу: https://hightech.in.ua/content/art-eclipse-platform (дата звернення 17.05.2021р). – Назва з екрана.

20. Eclipse як середовище розробки[Електронний ресурс] – Режим доступу: https://hightech.in.ua/content/art-eclipse-platform (дата звернення 17.05.2021р). – Назва з екрана.

21. Net beams як середовище розробки[Електронний ресурс] – Режим доступу: https://hightech.in.ua/content/art-netbeans-ide (дата звернення 17.05.2021р). – Назва з екрана.

ДОДАТКИ

Додаток А

Опис проекту у файлі 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>untitled</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>15</maven.compiler.source>
        <maven.compiler.target>15</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
```

Опис проекту у файлі pom.xml

```
            </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>
```