Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ ФРЕЙМВОРКІВ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-3
Фіалка Є.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Фіала Єгору Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та порівняння фреймворків для автоматизованого тестування вебзастосунків.

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 30 грудня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, данні з інтернет джерел, статті.

4. Перелік питань, що потрібно опрацювати в роботі

1. Провести огляд літератури та досліджень що до фреймворків для автоматизованого тестування.
2. Визначити основні критерії порівняння фреймворків.
3. Виконати аналіз фреймворків згідно з критеріями.
4. Провести практичні експерименти з використанням фреймворків.
5. Здійснити аналіз отриманих результатів.
6. Підготувати рекомендації щодо використання фрейморків

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Презентація, таблиці порівняння фреймворків, зображення результатів тестів, діаграми фреймворків.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	04.11.23-08.11.23	
3	Аналіз літератури з досліджуваної проблеми	09.11.23-12.11.23	
4	Аналіз технічних засобів	13.11.23-18.11.23	
5	Розробка методу	18.12.23-23.12.23	
6	Програмна реалізація	23.12.23-25.12.23	
7	Оформлення пояснювальної записки	25.12.23-30.12.23	
8	Перевірка на плагіат	03.12.2023	
9	Рецензування	05.12.2023	
10	Підготовка презентації та доповіді	16.12.2023	
11	Занесення роботи в електронний архів	09.01.2024	
12	Попередній захист кваліфікаційної роботи	09.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Тітова О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 10 рис., 5 табл., 38 джерел.

ПОРІВНЯННЯ,ФРЕЙМВОРК, ДОСЛІДЖЕННЯ, ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ, АВТОМАТИЗАЦІЯ, SERENITY, CYPRESS, ROBOT FRAMEWORK, REDWOODHQ, SAHI, GAUGE, CITRUS FRAMEWORK, KARATE-DSL, GALEN FRAMEWORK.

Об’єктом дослідження є вибір та порівняння фреймворків для автоматизованого тестування вебзастосунків у сучасному інформаційному середовищі з метою визначення їх переваг та недоліків. Вибрано та проаналізовано різні фреймворки, включаючи Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Galen Framework, Citrus Framework та Karate-DSL. Методологія дослідження передбачає визначення критеріїв порівняння, виконання тестових сценаріїв з використанням кожного фреймворку та аналіз результатів. Результати дослідження допоможуть обґрунтувати вибір найбільш підходящого фреймворку для конкретного проекту, підвищуючи ефективність та надійність тестування вебзастосунків. Кваліфікаційна робота має практичне значення для розробників та тестувальників вебзастосунків та сприяє розвитку інформаційної технології.

DEVELOPMENT, FRAMEWORK, RESEARCH, TESTING OF WEB ADD-ONS, AUTOMATION, SERENITY, CYPRESS, ROBOT FRAMEWORK, REDWOODHQ, SAHI, GAUGE, CITRUS FRAMEWORK, KARATE-DSL, GALEN FRAMEWORK.

The object of investigation is the selection and alignment of frameworks for automated testing of web sites in the current information environment by identifying their advantages and disadvantages. Selected and analyzed a variety of frameworks, including Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Galen Framework, Citrus Framework and Karate-DSL. The research methodology transfers the selected criteria to the evaluation, the integration of test scenarios from the skin framework and the analysis of the results. The research results will help to select the most suitable framework for a specific project, increasing the efficiency and reliability of testing web sites. A qualified job is of practical importance for web developers and testers and supports the development of information technology.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс для взаємодії між різними програмами або компонентами)

UI – User Interface (інтерфейс користувача, який надає зручний спосіб взаємодії з програмою)

ВСТУП

Сучасний світ програмного забезпечення вимагає високої якості вебзастосунків. Актуальність теми дослідження полягає в тому, що веб-застосуноки стали невід'ємною частиною повсякденного життя. Вони використовуються для комунікації, розваг та роботи в бізнесі. Висока актуальність полягає в тому, що вони стають складнішими та функціональнішими, а це, в свою чергу, ставить під сумнів якість та надійність вебзастосунків. Вебзастосунків постійно змінюються та розвиваються. Нові технології, стандарти та функціональні вимоги вимагають постійного оновлення та покращення процесів тестування. Фреймворки для автоматизованого тестування стають надзвичайно важливими для забезпечення якості вебзастосунків. Якісне тестування та швидкий випуск оновлень можуть допомогти відрізнятися на ринку та забезпечити конкурентну перевагу.

Об'єктом дослідження є фреймворки для автоматизованого тестування вебзастосунків, які є інструментами для полегшення процесу тестування вебзастосунків. Предметом дослідження є порівняння та аналіз цих фреймворків для визначення їхньої ефективності, переваг та недоліків.

Метою даної роботи є комплексне дослідження та порівняння різних фреймворків для автоматизованого тестування вебзастосунків. Результати дослідження дозволять визначити, які з фреймворків найбільше відповідають потребам веброзробників та тестувальників. Практична цінність полягає в можливості використовувати результати дослідження для покращення процесів автоматизованого тестування вебзастосунків та вибору оптимального фреймворку для конкретних завдань.

1 ОГЛЯД ЛІТЕРАТУРИ

1.1 Основні поняття та терміни

Введення до будь-якого наукового дослідження включає в себе осмислення та визначення ключових понять та термінів, які стануть фундаментом для подальших розділів та аналізу. Дана частина роботи присвячена розгляду основних понять та термінів, що використовуються у контексті автоматизованого тестування вебзастосунків та фреймворків, призначених для цієї мети.

Автоматизоване тестування: це процес використання програмних засобів для виконання тестів на вебзастосунках без прямого втручання користувача. Він дозволяє автоматизовано перевіряти функціональність, продуктивність та стабільність вебзастосунків.

Фреймворк для автоматизованого тестування: це набір інструментів, бібліотек та правил, які допомагають автоматизовано створювати та виконувати тести вебзастосунків. Фреймворки спрощують процес автоматизації, надаючи готові рішення для створення та виконання тестів.

Вебзастосунок: це програмне забезпечення, доступне через веббраузер, яке надає користувачам можливість взаємодіяти з різними функціональними можливостями, такими як сторінки, форми та інші елементи, через інтернет.

Тестування якості: процес визначення рівня якості програмного забезпечення, включаючи його функціональність, продуктивність та стабільність.

Продуктивність тестування: вимірювання швидкості, відгуків та інших параметрів роботи вебзастосунків під навантаженням.

Ці основні поняття і терміни є важливими для розуміння дослідження та порівняння фреймворків для автоматизованого тестування вебзастосунків.

Вони утворюють основу для подальшого аналізу та висновків, які будуть представлені в цій роботі.

1.2 Історія розвитку фреймворків для автоматизованого тестування

Історія розвитку фреймворків для автоматизованого тестування є важливою темою. Вона розпочалася після з'явлення перших комп'ютерів і розширилася разом із зростанням складності програмного забезпечення. У цьому розділі розглянемо основні етапи розвитку фреймворків для автоматизованого тестування та їх вплив на сучасну практику.

Перші спроби автоматизації тестування виникли на початку комп'ютерної ери. У 1947 році, під час розробки ENIAC, вважається, що Грейс Гоппер створила перший автоматизований тест для перевірки працездатності цього величезного обчислювального пристрою.

Починаючи з 1950-х років, з'явилися перші мови програмування, які використовувалися для створення тестових сценаріїв та фреймворків. Наприклад, мова SIPP (Symbolic Input to Program Production) була розроблена для створення автоматизованих тестів для UNIVAC комп'ютерів.

У 1990-х роках розвиток фреймворків для автоматизованого тестування почав набирати обертів. З'явилися перші спеціалізовані фреймворки, призначені для конкретних мов програмування та платформ.

З появою Інтернету та вебзастосунків, зростала потреба у фреймворках для тестування вебінтерфейсів. Selenium, розпочавши свій розвиток в 2004 році, став одним з популярних фреймворків для автоматизованого тестування вебзастосунків.

Сучасні фреймворки для автоматизованого тестування надають більше можливостей для модульного тестування, інтеграції з іншими інструментами

розробки та автоматизованого випуску програмного забезпечення. Це дозволяє створювати більш складні та потужні системи тестування.

За останні десятиліття відзначається поширення використання фреймворків з відкритим вихідним кодом, таких як Robot Framework, для забезпечення доступу до тестових інструментів для різних команд та розробників.

1.3 Попередні дослідження та порівняння фреймворків

Раніше проведені дослідження зосереджувалися на різних аспектах фреймворків для автоматизованого тестування вебзастосунків. Деякі з них вивчали ефективність фреймворків для конкретних мов програмування, встановлюючи, які з них найкраще підходять для певних завдань. Інші дослідження порівнювали фреймворки за їхніми можливостями для тестування різних типів вебзастосунків, таких як мультимедійні, динамічні або мобільні застосуноки.

Результати порівняння фреймворків з попередніх досліджень надають важливі відомості про їхні переваги та недоліки. Наприклад, деякі дослідження підкреслюють високу продуктивність та швидкість фреймворків, які базуються на власних мовах програмування, тоді як інші акцентують універсальність та гнучкість фреймворків з відкритим вихідним кодом.

Зокрема, стаття з інтернету, яка була включена у цей досліджуваний контекст, акцентує на актуальності фреймворків для модульного тестування в мові програмування Java [1].

Автори статті розглядають найбільш популярні фреймворки для модульного тестування та надають порівняльний аналіз двох з них, зокрема JUnit та TestNGX. Результати цього порівняльного аналізу дають важливі

відомості про те, як ці фреймворки відповідають на сучасні вимоги до модульного та компонентного тестування.

Основна увага статті приділяється функціональним можливостям фреймворків, групуванню тестових випадків, параметризації тестів та іншим аспектам, що становлять суттєвий внесок у розробку тестових сценаріїв та їх виконання. Реальний досвід використання фреймворків в статті робить порівняння ще більш об’єктивним та практично значущим [1].

Важливим доповненням до попередніх досліджень є погляд на актуальні питання щодо використання фреймворків в сучасних умовах розробки програмного забезпечення. Це включає вивчення питань масштабованості, можливостей розширення фреймворків та їх впливу на весь життєвий цикл розробки вебзастосунків [1].

Загальний контекст цих досліджень створює глибший уявлення про те, як вибір фреймворку може впливати на якість, продуктивність та розробку вебзастосунків у реальних проектах.

Наступна стаття, розглянута в даному контексті, зосереджується на двох інструментах, що представляють значний інтерес для тестувальників, зокрема в контексті написання end-to-end-тестів та UI-тестування.

Пошук оптимального інструменту для автоматизованого тестування не завжди є завданням простим. У цьому контексті, зіставлення можливостей Playwright та Selenium стосовно підтримки мов програмування та браузерів є важливим аспектом, який визначає ефективність та придатність кожного фреймворку для конкретних проєктів.

Playwright підтримує JavaScript/TypeScript, Java та Python, що робить його високоякісним вибором для широкого спектру розробників. Однак, стаття вказує на перевагу Selenium у сенсі підтримки декількох мов програмування, включаючи Java, Python, C#, Ruby і JavaScript. Це може виявитися критичним у випадках, коли розробка здійснюється командою різних фахівців з різних мов програмування.

Ще однією важливою характеристикою є підтримка різних браузерів. Тут Selenium виходить переможцем, оскільки він охоплює не лише Chrome, Edge, Chromium, та Webkit, а й Firefox, IE, Opera. Навіть якщо Playwright теж вказується як підтримуючий Firefox, стаття наголошує, що на практиці це може бути не завжди ефективним, порівняно із Selenium.

Важливою рисою порівняння є здатність працювати з різними версіями браузерів. Тут Selenium виявляється перевагою, адже він надає можливість запуску тестів на старших версіях браузера, що є критичним, наприклад, при тестуванні продуктів, де користувачі не завжди оновлюють свій браузер до останньої версії. Playwright обмежується останньою стабільною версією, що може бути недоцільним для деяких сценаріїв тестування.

Цінним аспектом порівняння є підхід до взаємодії з браузером. Selenium використовує зовнішні команди, що імітують дії тестувальника (наприклад, клік або введення тексту), тоді як Playwright взаємодіє з браузером внутрішньо, що призводить до вищої швидкості та більшого функціоналу. Останній може, наприклад, відкривати панель DevTools всередині браузера, що надає тестувальнику доступ до важливої інформації для відлагодження.

Додатково, Playwright вирізняється своєю різноманітністю засобів для звітності та аналізу тестів. За допомогою вбудованих функцій, таких як відео сесії, скриншоти та html-звіт, він спрощує процес аналізу тестових результатів. У Selenium для досягнення схожого функціоналу потрібно встановлювати та налаштовувати додаткові засоби.

Невідомо важливий фактор при порівнянні - робота із API. Playwright надає вбудовані функції для роботи з API-запитами, тоді як Selenium вимагає додаткових бібліотек для цієї мети. Це робить Playwright зручнішим вибором для тестування API-взаємодій, забезпечуючи зручні та швидкі засоби для їхнього покриття.

Обидва інструменти, Playwright і Selenium, використовуються для автоматизації тестування і мають схожий функціонал. Однак, стаття виокремлює важливі відмінності між ними, надаючи читачеві глибше

розуміння їхніх можливостей та призначення. Наприклад, вона підкреслює перевагу Playwright у спрощенні коду завдяки своїм функціям та готовим рішенням для таких завдань, як прокрутка сторінки до елементу. В той час як Selenium може вимагати кастомних методів для реалізації схожих завдань.

Стаття також стверджує, що обидва фреймворки не обмежують тестувальників у виборі архітектури тестів. Тестувальник може вибрати той підхід, який найбільше відповідає потребам його проєкту, використовуючи доступні функції фреймворку. Таке визначення свободи у виборі підходу до написання тестових сценаріїв стає важливим фактором при розгляді ефективності та зручності використання цих інструментів.

Одним з ключових висновків статті є те, що Playwright, з його широким функціоналом та підтримкою усіх основних браузерів, стає обіцяючим інструментом для end-to-end-тестів та API тестування. Розглядаючи його зручності та надійність, автори статті визначають його як потенційно кращий вибір для сучасних проєктів.

Поруч із цим, стаття вказує на переваги Playwright у порівнянні з іншими популярними інструментами, такими як Cypress, розглядаючи його як більш розвинений та стійкий інструмент для автоматизованого тестування. Ця порівняльна характеристика вносить важливий внесок у вибір тестувальників та розробників щодо найкращого інструменту для їхніх конкретних потреб.
Усі ці аспекти порівняльного аналізу Playwright та Selenium розширюють знання про можливості та переваги цих фреймворків у сфері автоматизованого тестування веб-додатків, надаючи читачеві більше інформації для прийняття обгрунтованих рішень щодо вибору найбільш підходящого інструменту для його конкретного проєкту [2].

Незважаючи на розгортану роботу в цій галузі, є прогалини та невирішені питання, які потребують подальших досліджень. Зокрема, важливо вивчити вплив фреймворків на розробку, тестування та випуск вебзастосунків в реальних проектах. Також, дослідники починають досліджувати аспекти

масштабованості та можливості розширення фреймворків, щоб вони могли відповідати вимогам сучасної розробки.

1.4 Постановка задачі дослідження

Постановка задачі дослідження є важливим етапом в розробці кваліфікаційна роботи. Задачі дослідження визначають обсяг та напрямок роботи, вказують на конкретні цілі, які дослідник планує досягти під час проведення свого дослідження. У цьому розділі будуть сформульовані та детально розглянуті завдання, які мають бути вирішені в процесі аналізу та порівняння фреймворків для автоматизованого тестування вебзастосунків.

Мета дослідження: головною метою даної роботи є проведення аналізу та порівняння різних фреймворків для автоматизованого тестування веб-застосунків з метою визначення їх переваг та недоліків. Основною метою є обґрунтування вибору оптимального фреймворку для конкретного проекту на практиці.

Завдання дослідження: для досягнення визначеної мети необхідно вирішити наступні завдання:

Провести огляд літератури та наявних досліджень щодо фреймворків для автоматизованого тестування вебзастосунків для збирання інформації щодо їхніх характеристик та особливостей.

Визначити основні критерії, за якими будуть порівнюватися фреймворки, такі як ефективність тестування, функціональність, підтримка мов програмування, підтримка різних браузерів та платформ, розширення та плагіни, зручність використання та наявність документації.

Виконати аналіз обраних фреймворків відповідно до визначених критеріїв та підготувати порівняльну таблицю результатів.

Провести практичні експерименти з використання обраних фреймворків для автоматизованого тестування вебзастосунків на прикладі реального проекту.

Зробити аналіз отриманих результатів та зробити висновки щодо кращого вибору фреймворку для конкретного проекту.

Підготувати рекомендації щодо використання обраного фреймворку та можливості подальшого його розвитку.

Завдання дослідження ретельно покривають всі аспекти, які необхідно враховувати під час аналізу та вибору фреймворку для автоматизованого тестування веб-додатків. Вони допоможуть досягти поставленої мети та визначити оптимальний вибір для конкретного проекту.

2 ФРЕЙМВОРКИ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ

2.1 Обгрунтування вибраних фреймворків

Обрання правильних фреймворків для автоматизованого тестування вебзастосунків є критично важливим завданням у розробці програмного забезпечення. Першим кроком у виборі фреймворків було визначення завдань дослідження. Основне концентрування на дослідження та порівняння фреймворків для автоматизованого тестування вебзастосунків, зокрема їхніх можливостей, продуктивності та гнучкості.

Популярність та активність спільноти розробників є важливими факторами при виборі фреймворків. Популярні фреймворки зазвичай мають ширший спектр підтримуваних можливостей та велику кількість доступних розширень та плагінів.

Для дослідження та порівняння фреймворків, необхідно мати фреймворки, які добре поєднуються з іншими інструментами розробки, такими як інструменти контінеризації, системи управління версіями та інші.

Оскільки розробка вебзастосунків може використовувати різні мови програмування та технології, важливо вибрати фреймворки, які підтримують широкий спектр мов та технологій.

Також враховується попередні дослідження та рекомендації в галузі автоматизованого тестування вебзастосунків, які допомогли визначити фреймворки, які вже зарекомендували себе в реальних проектах.

Тому було обрано наступні фреймворки:

– Serenity

– Cypress

– Robot Framework

– RedwoodHQ

– Sahi

– Gauge

– Galen Framework

– Citrus Framework

– Karate-DSL

2.2 Огляд вибраних фреймворків

2.2.1 Serenity

Serenity – це високорівневий фреймворк для автоматизованого тестування вебзастосунків. Він володіє рядом унікальних особливостей та можливостей, які роблять його важливим інструментом для тестування вебзастосунків [6].

Основні можливості Serenity:

– Розширена система звітності: Serenity надає докладну систему звітності, яка дозволяє створювати докладні звіти про результати тестів. Це полегшує відстеження та аналіз результатів тестування.

– Жива документація: фреймворк підтримує створення живої документації, що дозволяє створювати зрозумілі та доступні описи тестових сценаріїв. Це полегшує спільну роботу між розробниками та тестувальниками.

– Можливості для автоматизації вебзастосунків: Serenity дозволяє автоматизувати тестування вебзастосунків шляхом написання тестів на основі ключових слов і використання вбудованих інструментів для взаємодії з вебсторінками.

– Широкий вибір мов програмування: фреймворк підтримує кілька мов програмування, включаючи Java, Kotlin, та інші. Це дає розробникам можливість вибору мови, з якою вони найкраще знайомі та зручно працюють.

– Розширені можливості для тестування мультимедійних та динамічних вебзастосунків: Serenity відзначається високою продуктивністю та можливістю тестування різних типів вебзастосунків, включаючи мультимедійні та динамічні застосуноки.

– Інтеграція з іншими інструментами: Serenity може бути інтегрований з різними іншими інструментами розробки, такими як Maven, Gradle, JIRA, Jenkins, та інші.

Основні компоненти Serenity включають Serenity BDD ща дозволяє писати та організовувати тести в стилі Behavior-Driven Development (BDD) та Serenity Reports що адає зручні та інтерактивні звіти про виконання тестів [3].

Нижче реалізація коду базової структури тестування з використанням Serenity на прикладі сторінки google.

Лістинг 2.1 Створення тесту:

```
public class MyTest {
    @Managed
    WebDriver driver;
    @Steps
    MySteps mySteps;
    @Test
    public void sampleSerenityTest() {
        // Відкриття веб-сторінки
        mySteps.openHomePage();
        // Виконання дій на сторінці
        mySteps.performSomeActions();
        // Перевірка результатів
        mySteps.verifyResults();
          }
}
```

Лістинг 2.2 Створення кроків:

```
// Створення Кроків (Steps)
public class MySteps {
    @Step
    public void openHomePage() {
        // Код для відкриття веб-сторінки
        // URL веб-сторінки Google
        String googleUrl = "https://www.google.com";
        // Відкриття веб-сторінки за допомогою WebDriver
        driver.get(googleUrl);    }
    @Step
    public void performSomeActions() {
        // Код для виконання дій на сторінці
        // Натискання на кнопку або елемент
        clickOnElement(By.id("buttonId"));
        // Заповнення форми
        enterTextIntoElement(By.name("username"), "testuser");
        enterTextIntoElement(By.name("password"), "password123");
        // Відправлення форми
        submitForm(By.id("loginForm"));    }
    // Метод для натискання на елемент
    private void clickOnElement(By locator) {
```

```
    getDriver().findElement(locator).click();    }
// Метод для введення тексту в елемент
private void enterTextIntoElement(By locator, String text) {
    getDriver().findElement(locator).sendKeys(text);    }
// Метод для відправлення форми
private void submitForm(By formLocator) {
    getDriver().findElement(formLocator).submit();   }
@Step
public void verifyResults() {
    // Код для перевірки результатів
    // Перевірка тексту на елементі
    String actualText = getTextFromElement(By.id("resultElement"));
    assertThat(actualText).isEqualTo("Expected Text");
    // Перевірка наявності елемента
          boolean elementPresent =
      isElementPresent(By.className("resultClass"));
    assertThat(elementPresent).isTrue();    }
// Метод для отримання тексту з елемента
private String getTextFromElement(By locator) {
    return getDriver().findElement(locator).getText(); }
// Метод для перевірки наявності елемента
private boolean isElementPresent(By locator) {
    return !getDriver().findElements(locator).isEmpty();    }}
```

Лістинг 2.3 Створення звіту:

```
@RunWith(SerenityRunner.class)
@Narrative(title = "Звіт про тестування веб-додатка", text = { "Цей звіт містить результати тестування веб-додатка." })
public class MySerenityTest {
    @Managed
    WebDriver driver;
    @Test
    @Title("Перевірка відкриття головної сторінки")
    @WithTags({
        @WithTag("feature:homepage"),
        @WithTag("test:smoke")   })
    public void verifyHomePage() {
      // тестові дії тут   }
    @Test
    @Title("Перевірка виконання деяких дій на сторінці")
    @WithTags({
        @WithTag("feature:actions"),
        @WithTag("test:regression")    })
    public void verifyActionsOnPage() {
      // тестові дії тут    }}
```

Цей приклади демонструює базову структуру тестування з використанням Serenity з використовуванням язика програмування Java.

Після виконання тестів, Serenity автоматично згенерує звіт, який можна переглянути за допомогою Serenity Dashboard, приклад інтерфейсу зображено нижче (рис. 2.1). Звіт буде містити інформацію про пройдені тести, їх заголовки, теги, а також іншу корисну інформацію.

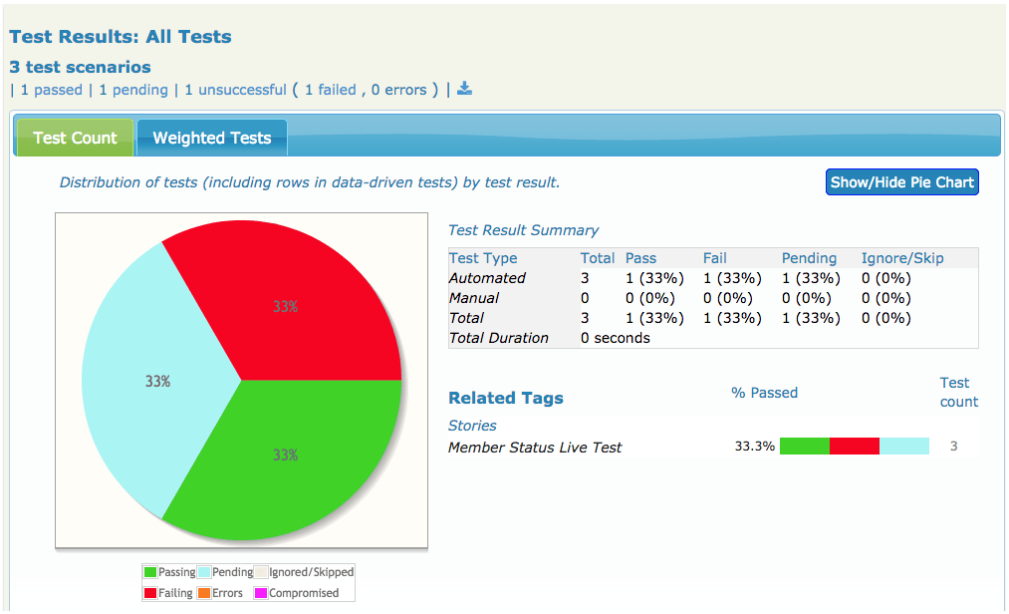

Рисунок 2.1 – Приклад інтерфейсу Serenity Dashboard

Serenity є потужним фреймворком для автоматизованого тестування вебзастосунків, який надає велику кількість можливостей для створення ефективних та надійних тестів.

2.2.2 Cypress

Cypress – це платформа автоматизації тестування, яка відповідає принципам TDD і орієнтована на розробника. Це спрощує процес наскрізного тестування, робить його зручнішим і простішим. Що відрізняє Cypress від Selenium, так це його архітектура. У той час як Selenium WebDriver працює поза браузером, Cypress працює безпосередньо в браузері. Цей підхід забезпечує узгоджені результати та дозволяє отримати доступ до будь-якого об’єкта, не турбуючись про серіалізацію об’єкта чи наскрізні протоколи. По суті, ви запускаєте свою програму в Cypress, що дозволяє інфраструктурі миттєво сповіщати вас про все, що відбувається в браузері, і надає вам доступ

до кожного елемента об’єктної моделі документа (DOM). Крім того, Cypress полегшує налагодження програми, а розробники можуть використовувати інші інструменти протягом усього процесу розробки [7-8].

Розглянемо деякі базові кроки для написання тестів з використанням Cypress на мові програмуванні JavaScript.

Лістинг 2.4 Створення тесту Cypress:

```
describe('My First Test', () => {
  it('Visits the Cypress Test Page', () => {
  // Відкриваємо веб-сторінку
cy.visit('https://example.cypress.io');
// Перевіряємо, чи існує текст 'type' та натискаємо на нього
cy.contains('type').click();
  // Перевіряємо, чи URL містить '/commands/actions'
  cy.url().should('include', '/commands/actions');  });});});
```

Додамо асинхронні операції в наш тест, Cypress обробляє їх автоматично.

Лістинг 2.5 Додавання асинхронних операцій та асерцій:

```
// cypress/integration/sample_spec.js
describe('My Second Test', () => {
  it('Fills out a form', () => {
    // Відкриваємо веб-сторінку
    cy.visit('https://example.cypress.io');
    // Натискаємо на посилання з текстом 'forms'
    cy.contains('forms').click();
    // Вибираємо текстове поле за ім'ям 'email' та вводимо текст
    cy.get('[name="email"]').type('test@example.com');
```

```
// Вибираємо текстове поле за ім'ям 'password' та вводимо текст
cy.get('[name="password"]').type('password123');
// Позначаємо чекбокс
cy.get('[type="checkbox"]').check();
// Вибираємо перший радіобатон і його перевіряємо
cy.get('[type="radio"]').first().check();
// Вибираємо елемент з тегом 'select' та обираємо опцію 'Option 1'
cy.get('select').select('Option 1');
// Натискання на кнопку для відправки форми
cy.get('[type="submit"]').click();
// Перевірка, чи URL містить '/commands/actions'
cy.url().should('include', '/commands/actions');  }); });
```

Цей тест виконує декілька дій на сторінці та перевіряє, чи вони виконані успішно. Після запуску за допомогою команди “npx cypress run”, в терміналі виводиться інформацію про виконані тести та їх результати. Приблизний вигляд виводу зображенно на рисунку (рис. 2.2).

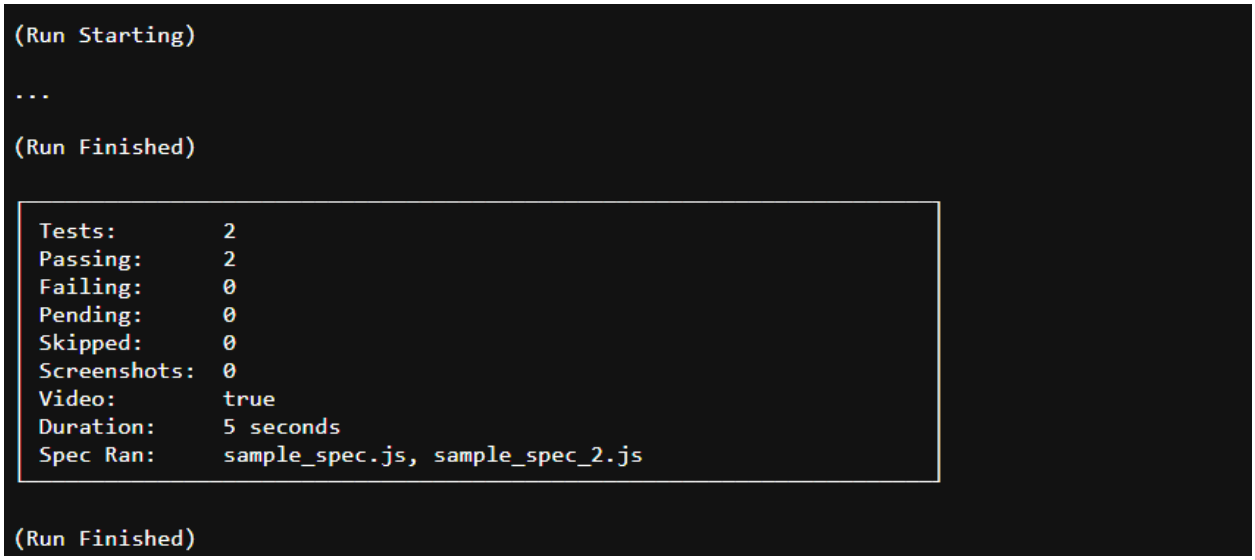

Рисунок 2.2 – Приклад консольного інтерфейсу cypress

2.2.3 Robot Framework

Robot Framework, розроблений спеціально для тестування, може бути цінним ресурсом. Він базується на ключових словах і робить створення та аналіз тестів більш зручними. Robot Framework надає доступ до багатьох бібліотек і інструментів для безкоштовного використання. Хоча його створено на Python, його можна успішно використовувати разом із Java та IronPython (.NET).

Хоча Selenium WebDriver є однією з найпопулярніших бібліотек для вебтестування, Robot Framework надає можливість тестувати не тільки вебсайти, але й інші об'єкти, такі як FTP, MongoDB, Android і Appium. Крім того, завдяки розширеному API, це потужний інструмент автоматизації, який можна легко розширити [11].

Розглянемо приклад написання тестів з використанням фреймворку для автоматизованого тестування веб-додатків Robot Framework.

Лістинг 2.6 Код файлу тестового сценарію «web_test.robot»:

```
*** Settings ***
Library         SeleniumLibrary
*** Variables ***
${BROWSER}      chrome
${URL}          https://www.example.com
*** Test Cases ***
Open Browser and Visit Example
    Open Browser    ${URL}    ${BROWSER}
    Title Should Be    Example Domain
Enter Data and Verify
    Input Text    name=username    my_username
    Click Button    name=submit_button
    Page Should Contain    Welcome, my_username
```

Після запуску цього тестового сценарію командою «web_test.robot» результати виконання виводяться у терміналі (рис. 2.3).

```
==============================================================================
Web Test
==============================================================================
Open Browser and Visit Example                                        | PASS |
------------------------------------------------------------------------------
Enter Data and Verify                                                 | PASS |
------------------------------------------------------------------------------
Web Test                                                               | PASS |
2 critical tests, 2 passed, 0 failed
2 tests total, 2 passed, 0 failed
==============================================================================
Output:  /path/to/output.xml
Log:     /path/to/log.html
Report:  /path/to/report.html
```

Рисунок 2.3 – Результати виконання тесту Robot Framework

Цей звіт показує, що обидва тести “Open Browser and Visit Example” та “Enter Data and Verify” були успішно виконані. Robot Framework використовує синтаксис, який легко читається і зрозуміти, і відмінно підходить для тестування веб-додатків.

2.2.4 RedwoodHQ

Цей інструмент створює вебінтерфейс, який дозволяє кільком тестувальникам співпрацювати та запускати тести з єдиного доступного місця. RedwoodHQ підтримує написання тестів на Java/Groovy, Python і C# для тестування вебзастосунків за допомогою Selenium, API або баз даних. Він

пропонує інтегроване середовище розробки та включає власну інтеграцію Git [12].

Особливістю RedwoodHQ є ключові слова, які полегшують створення та редагування тестових сценаріїв. Щоб створити тестовий сценарій, просто знайдіть потрібну дію, перетягніть її в тестовий приклад і вкажіть очікувані параметри.

Інтегроване середовище розробки в RedwoodHQ дозволяє створювати та редагувати тестові випадки та дії, а також виконувати тести. Також є можливість запускати паралельні тести та переглядати історію всіх попередніх тестів.

RedwoodHQ пропонує простий у використанні набір інструментів автоматизації тестування та містить багато вбудованих функцій, які можна використовувати для ефективного автоматизованого тестування [13].

Для прикладу напишемо тест який перевіряє, чи можна відкрити веб-сторінку "www.google.com" , з використаням мови програмування python з допомогою фреймворку RedwoodHQ.

Лістинг 2.6 Коду тесту, який перевіряє, чи можна відкрити веб-сторінку:

```
class MyTest(redwoodhq.TestCase):
   def test_open_page(self):
      self.driver.get("https://www.google.com") # Перевіряємо, чи відкрита
            self.assertEqual(self.driver.title, "Google")# веб-сторінка
MyTest().run()
```

Після запуску файлу, в консолі повідомляться що тест пройшов успішно (рис.2.4).

```
[INFO] Starting RedwoodHQ test suite
[INFO] Executing test case MyTest.test_open_page
[INFO] Test case MyTest.test_open_page passed
[INFO] Finished RedwoodHQ test suite
```

Рисунок 2.4 – Результати виконання тесту з RedwoodHQ

2.2.5 Sahi

Фреймворк Sahi пропонує як відкриту, так і професійну версію. Він діє як проксі-сервер, який можна інтегрувати в браузер. Ви можете використовувати панель інструментів Sahi, щоб запустити браузер для тестування.

Sahi забезпечує реєстрацію помилок і створення скріншотів кожного разу, коли ви взаємодієте з програмою. Наприклад, коли ви наводите курсор на елемент у браузері, Sahi надає список усіх доступних дій, які можна виконати з цим елементом.

Також можна відтворити сценарій за допомогою контролера. Функція запису та відтворення Sahi спрощує процес автоматизації тестування простих програм HTML. Однак варто зазначити, що покладатися виключно на запис і відтворення для створення надійних автоматизованих тестів, які можна підтримувати, не є найкращою практикою. Ця функція може бути корисною для початкового створення тестового випадку, але може вимагати подальшого вдосконалення для досягнення надійності та стійкості [15-17].

Для прикладу реалізації простого тесту з допомогою Sahi, створено тест на мові програмування JavaScript, який перевіряє чи можна успішно авторизуватися на сайті.

Лістинг 2.7 Код реалізації тесту з використанням фреймворку Sahi:

```
function test_login() {
  // Перейти на сторінку авторизації
  sahi.browser.goto("https://mysite.com/login");
  // Ввести ім’я користувача
  sahi.browser.setValue("username", "my_username");
  // Ввести пароль
  sahi.browser.setValue("password", "my_password");
  // Натиснути кнопку входу
```

```
sahi.browser.click("//input[@type='submit']");
// Перевірити, чи успішно авторизовано
assert.equal(sahi.browser.getTitle(), "Мій особистий кабінет");}
```

Після запуску тесту командою «sahi run» в консолі з’являється повідомлення про успішне виконання тесту (рис. 2.5) або що тест завершився невдачею (рис. 2.6).

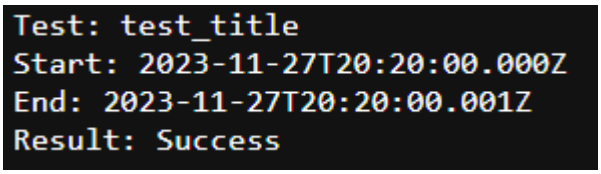

Рисунок 2.5 – Успішне виконання тесту.

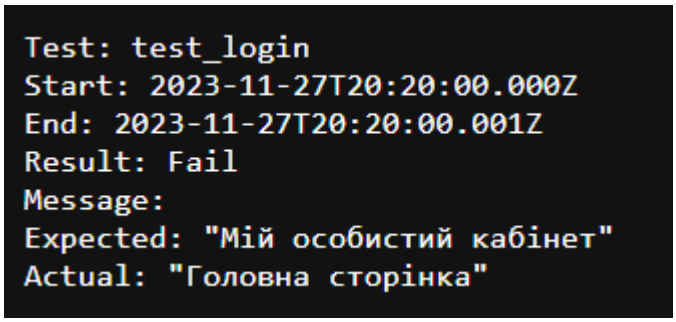

Рисунок 2.6 – Провальне виконання тесту.

2.2.6 Gauge

Gauge – це фреймворк для автоматизованого тестування, який надає ряд інноваційних можливостей для створення та виконання тестів вебзастосунків. Його особливістю є простота та легкість використання, а також підтримка кількох мов програмування [19].

Основні особливості Gauge:

– мовно-незалежний підхід: Gauge дозволяє тестувальникам використовувати різні мови програмування для написання тестів. Це важливо для команд, які мають розробницьку експертизу в різних мовах;

– простий синтаксис: фреймворк використовує простий та зрозумілий синтаксис для написання тестів, що полегшує їхнє створення та обслуговування;

– велика кількість плагінів: Gauge підтримує велику кількість розширень та плагінів, які розширюють його функціональність та дозволяють інтегрувати з іншими інструментами розробки;

– підтримка BDD (поведінково-орієнтованого розробки): Gauge підтримує підхід BDD, який дозволяє писати тести в формі природної мови і зближує тестувальників, розробників та бізнес-аналітиків;

– нтеграція з іншими інструментами: Gauge може бути легко інтегрований з іншими інструментами розробки, такими як Maven, Gradle, Jenkins, та інші;

Далі виконана реалізація тестового сценарію для перевірки відкриття сторінки Google за допомогою фреймворку Gauge та мови програмування Java з використанням Selenium.

Лістинг 2.8 Файл сценарію google_search.spec:

```
# google_search.spec
## Google Search Scenario
* Open browser to https://www.google.com
* Verify the title is "Google"
```

Лістинг 2.9 Код файлу реалізації кроків:

```
//Іморт необхідних бібліотек
import com.thoughtworks.gauge.Step;
import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
// Клас, який містить реалізацію кроків тестових сценаріїв для Gauge
public class StepImplementation {
// Об'явлення об'єкту WebDriver для взаємодії з браузером
   private WebDriver driver;
// Крок для відкриття браузера та переходу на вказану сторінку
   @Step("Open browser to <url>")
   public void openBrowser(String url) {
// Налаштування параметрів браузера (Chrome у даному випадку)
       ChromeOptions options = new ChromeOptions();
// Ініціалізація об'єкту WebDriver з параметрами браузера
       driver = new ChromeDriver(options);
       driver.get(url);
   }
   @Step("Verify the title is <title>")
   public void verifyTitle(String title) {
       assert driver.getTitle().equals(title);
       driver.quit();    }}
```

Запуск програми командою «gauge run» відкриє браузер, перейде на сторінку Google та перевірить, чи заголовок сторінки відповідає "Google". Та вивід в терміналі, що вказує на результати виконання тестів (рис.2.7). Зазвичай вивід містить інформацію про кількість успішних та невдалих кроків, час виконання тестів, а також іншу корисну інформацію.

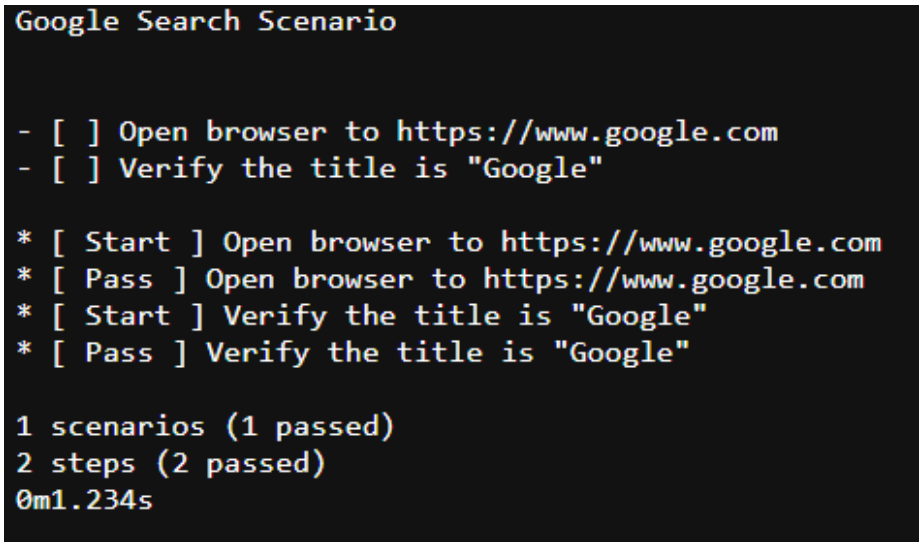

Рисунок 2.7 – Термінал після виконання тестів.

Gauge надає гнучкість та можливість вибору мови програмування для створення тестів. Він відзначається простотою використання та розширеністю функціональності завдяки великій кількості плагінів. У наступних розділах буде порівняння Gauge з іншими вибраними фреймворками та аналіз його переваги та обмеження [20].

2.2.7 Galen Framework

Galen Framework – це потужний інструмент для автоматизованого тестування вебзастосунків, який спеціалізується на перевірці макета та відповідності вебсайтів заданим правилам. Його основна функція полягає в тому, щоб вебзастосунок правильно відображався на екранах різних розмірів і в браузерах [29].

Важливо відзначити, що тести, які виконуються за допомогою цього інструменту, автоматично створюють детальні звіти HTML, включаючи знімки екрана для легкого порівняння результатів. Ці звіти також містять зображення для візуального порівняння макета з очікуваннями, що значно полегшує аналіз результатів тестування [26].

Для ознайомлення з фреймворком Galen Framework було реалізовано тест перевірки вигляду сторінки Google на робочому столі з розміром екрану 1200x800 пікселів.

Лістинг 2.9 текстовий файл з розширенням .test:

```
# google.test
= Homepage on desktop device
   @onDesktop
   http://www.google.com
      width: 1200px
      height: 800px
   body
      width: 100% of screen/width
   q # Перевірка, що пошукове поле знаходиться по центру екрану
      center-of: screen
      below: 50px
```

Після запуску тесту командою «galen test google.test --htmlreport reports» виконається тест і створиться звіт HTML у каталозі reports. У цьому звіті буде інформація (рис. 2.8), чи пройшов тест та які саме перевірки були виконані.

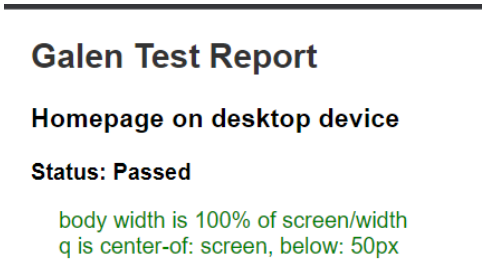

Рисунок 2.8 – HTML звіт виконання тестів.

Це базовий приклад використання Galen Framework для тестування вигляду веб-сторінок. Гален дозволяє вам визначити свої власні правила та перевіряти їх виконання для різних розмірів екрану.

2.2.8 Citrus Framework

Citrus Framework – це інноваційна автоматизована платформа тестування вебзастосунків, яка спеціалізується на тестуванні взаємодії між програмою та зовнішніми службами. Ця структура дозволяє створювати ефективні та надійні тести для перевірки взаємодії вашої програми з іншими системами та службами [21].

Ключові особливості Citrus Framework:

– Декларативний синтаксис: Citrus використовує декларативний синтаксис для опису тестів і робить їх читабельними та зрозумілими. Це дозволяє тестувальникам швидко створювати тести та керувати ними.

– Тестування взаємодії сервісів: фреймворк спеціалізується на тестуванні взаємодії між вашою програмою та зовнішніми службами, такими як служби REST, служби SOAP, JMS та інші. Ви можете створювати тести для перевірки обміну даними та поведінки програми під час взаємодії з цими службами.

– Вбудований механізм імітації: Citrus надає вбудований механізм імітації, який дозволяє імітувати зовнішні служби під час тестування. Це полегшує створення ізольованих тестів і налагодження взаємодії між додатком і службами.

– Підтримка різних протоколів: Citrus підтримує різні протоколи взаємодії, що дозволяє тестувати різні аспекти взаємодії між додатком і зовнішніми службами.

Для більш детального дослідження фреймворку було реалізовано приклад тесту Citrus для взаємодії зі сторінкою пошуку Google, введення пошукової інформації та натискання кнопки пошуку [24].

Лістинг 2.10 Код тесту Citrus мовою програмування java:

```
import com.consol.citrus.annotations.CitrusTest;
import com.consol.citrus.dsl.design.TestDesigner;
import com.consol.citrus.dsl.testng.TestNGCitrusTest;
import com.consol.citrus.selenium.actions.SeleniumActionBuilder;
import org.openqa.selenium.By;
import org.testng.annotations.Test;
public class GoogleSearchIT extends TestNGCitrusTest {
    @Test
    @CitrusTest
    public void testGoogleSearch() {
        // Визначте термін пошуку
        String searchTerm = "Citrus Framework";
        // Запустіть дизайнер тестів
        TestDesigner testDesigner = new TestDesigner();
        // Запустіть браузер та відкрийте Google
      testDesigner.selenium().start().browser().url("https://www.google.com");
        // Введіть термін пошуку
       testDesigner.selenium().find().element(By.name("q")).send(searchTerm);
        // Натисніть кнопку пошуку
        testDesigner.selenium().find().element(By.name("btnK")).click();
        // Перевірте, що сторінка з результатами пошуку
відображається
        testDesigner.selenium().find().element(By.id("search")).exists();
        // Зупинка браузера
        testDesigner.selenium().stop();
        // тест
        testDesigner.execute();   }}
```

Після запуску цього тесту за допомогою TestNG браузер запущений і відкрито веб-сайт Google, у полі пошуку буде введено термін "Citrus Framework" натиснута кнопка пошуку та відображатися сторінка з результатами пошуку.

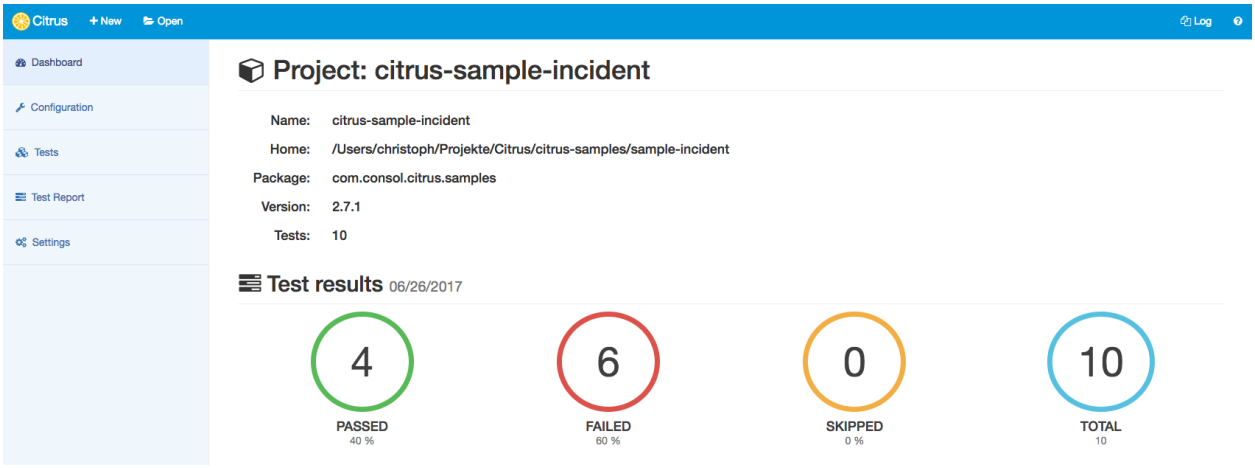

Рисунок 2.9 – Приклад результату виконання тестів за інтерфейсу "Citrus Framework".

Citrus Framework є важливим інструментом для тестування взаємодії програми з іншими системами та службами. Цей фреймворк дозволяє створювати ефективні та детальні тести для перевірки правильності роботи вашої програми під час взаємодії з різними службами.

2.2.9 Karate-DSL

Karate-DSL – це автоматизована платформа тестування REST API, яка може бути ідеальним вибором для тих, хто вже використовує Cucumber і потребує тестування REST API. Цей фреймворк існує вже більше року і щодня завойовує все більше прихильників серед тестувальників API.

Ключові особливості Karate DSL:

– Інтеграція з Cucumber JVM: Karate DSL базується на Cucumber JVM і дозволяє використовувати знайомі функції для створення та запуску тестів, перегляду результатів і створення звітів;

– синтаксис Gherkin: у структурі використовується синтаксис Gherkin, що робить його легким для читання та розуміння всією командою проекту;

– підтримка REST API: Karate-DSL спеціалізується на тестуванні REST API і надає зручні інструменти для визначення та виконання тестів на різних рівнях взаємодії API;

– розширені звіти. Платформа генерує докладні й інформативні звіти про результати тестування, які допомагають вам швидко виявити проблеми та налагодити тести.

Розглянемо основи роботи з Karate DSL для автоматизованого тестування веб-додатків. Для цього реалізовано приклад тесту.

Лістинг 2.11 Файл з тестовим сценарієм з розширенням “.feature”:

```
Feature: Google Search

  Background:
    * configure driver = { type: 'chrome' }

  Scenario: Search for "Karate DSL"
    Given url 'https://www.google.com'
    And input('input[name="q"]', 'Karate DSL')
    When submit('input[name="btnK"]')
    Then waitFor('input[name="q"]').input == 'Karate DSL'
    And match text('#search h3', 'Karate DSL - GitHub')
```

Лістинг 2.11 Запуск карате-тесту “.java”:

```
import com.intuit.karate.junit5.Karate;
class GoogleSearchTest {
   @Karate.Test
   Karate testGoogleSearch() {
      return Karate.run("google-search").relativeTo(getClass());
   }
}
```

Після виконання тесту за допомогою Karate DSL, результати відображаються у форматі BDD в консолі (рис.2.10).

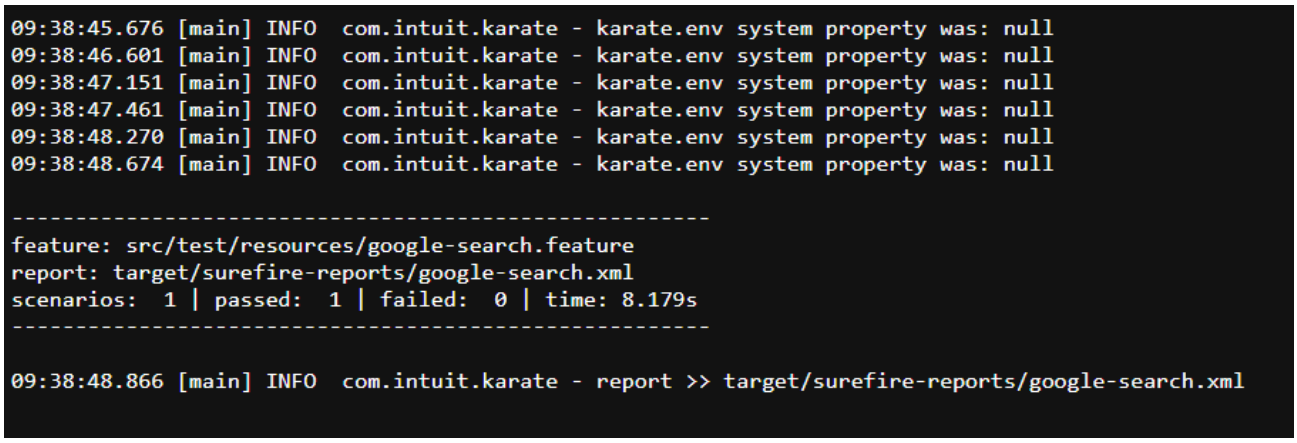

Рисунок 2.10 – Результат виконання тесту Karate DSL

Karate-DSL – це потужний інструмент для тестування REST API, особливо для тих, хто вже використовує Cucumber і просто хоче розширити свої можливості до рівня тестування API

2.3 Характеристики та особливості фреймворків

Розглянемо характеристики та особливості кожного з обраних фреймворків для автоматизованого тестування вебзастосунків. Кожен з цих фреймворків має свої унікальні особливості, що роблять його особливим та корисним інструментом для тестування.

– Синтаксис та мови програмування: кожен фреймворк використовує свій власний синтаксис та може підтримувати різні мови програмування. Проаналізуємо, наскільки легко вчити та використовувати цей фреймворк, а також які мови програмування він підтримує;

– підтримка різних браузерів та платформ: розглянемо, наскільки ефективно кожен фреймворк підтримує різні браузери та платформи. Це важливо для забезпечення, що тести працюють коректно на різних конфігураціях;

– можливості розширення: дослідимо можливості розширення кожного фреймворка, включаючи наявність плагінів та інструментів для створення додаткових функцій та можливостей;

– зручність використання: оцінимо, наскільки фреймворки є зручними та дружелюбними для користувача, як легко встановити та налаштувати їх, та як вони підтримують процес розробки та тестування;

– підтримка спільноти та документація: огляд також включатиме аналіз підтримки спільноти та доступності документації для кожного фреймворку.

Основною важливаю спільною рисою усіх фреймворків є доступність, усі обрані для порівняння фреймворків з відкритим кодом. Що надає змогу для перегляду та ретельного вивчення, дозволяє переконатися у відсутності вразливостей та неприйнятних для користувача функцій.

3 ПОРІВНЯННЯ ФРЕЙМВОРКІВ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ

3.1 Вибір критеріїв, методів та інструментів для дослідження та порівняння фреймворків.

Був зроблен вибір на аспектах по яких буде проводитись детальне порівняння обраних фреймворків для автоматизованого тестування вебзастосунків. Метою цього порівняння є визначення переваг і недоліків кожного з фреймворків, за певними критеріями та методами. Ці аспекти є актуальними та важливими в контексті дослідження і порівняння фреймворків для автоматизованого тестування вебзастосунків.

Розглянемо наступні аспекти:

– ефективність тестування: оцінимо, наскільки ефективно кожен фреймворк дозволяє створювати та виконувати тести для вебзастосунків. Це включає в себе швидкість виконання тестів та зручність створення нових тестів. Швидкість та зручність виконання тестів є критичними для забезпечення швидкого і надійного тестування вебзастосунків. Це важливо для проектів з короткими циклами розробки та постійними змінами в додатку;

– функціональність: розглянемо, які функціональні можливості надає кожен фреймворк, включаючи можливість тестування різних аспектів вебзастосунків, таких як взаємодія з елементами інтерфейсу, робота з формами, перевірка валідації даних тощо. Важливо мати можливість тестувати всі аспекти вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних та інші. Фреймворк повинен надавати засоби для відтворення реальних сценаріїв взаємодії користувача з додатком;

– підтримка мов програмування: які мови програмування підтримуються кожним фреймворком та наскільки легко програмувати тести в цих мовах.

Різні команди та проекти можуть використовувати різні мови програмування. Важливо мати можливість працювати з фреймворком у мові, з якою команда зручно володіє та яка найкраще підходить для конкретного проекту;

– підтримка різних браузерів та платформ: наскільки ефективно кожен фреймворк підтримує різні браузери та платформи для тестування вебзастосунків. Вебзастосунки повинні підтримувати різні браузери та платформи. Фреймворк для тестування повинен надавати можливість виконання тестів на різних конфігураціях;

– розширення та плагіни: можливості розширення кожного фреймворку, включаючи наявність плагінів та інструментів для розширення функціоналу. Зручність розширення функціоналу фреймворку може бути корисною для адаптації до конкретних потреб проекту.

Зручність використання та навчання: наскільки фреймворки є зручними та дружелюбними для користувача, а також як легко вчити та працювати з ними. Фреймворк повинен бути дружелюбним для користувача, щоб зменшити час, необхідний для навчання та роботи з ним.

– Підтримка спільноти та документація: аналіз підтримки спільноти та доступності документації для кожного фреймворку. Наявність активної спільноти та якісної документації може значно спростити роботу з фреймворком та вирішення проблем.

Усі ці аспекти допоможуть об’єктивно порівняти фреймворки та визначити, який з них найкраще відповідає потребам в автоматизованому тестуванні вебзастосунків.

3.2 Порівняння фреймворків за різними критеріями

Ефективність тестування – це важливий аспект, який допомагає визначити, наскільки добре фреймворки для автоматизованого тестування вебзастосунків виконують свої функції. Оцінюючи ефективність тестування, розглядається кілька ключових аспектів: швидкість виконання тестів, зручність створення нових тестів.

Порівняємо фреймворки Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Galen Framework, Citrus Framework, Karate-DSL за ключивими аспектами показано у таблиці 3.1

Таблиця 3.1 – Порівняння ефективністі тестування

Ефективність тестування									
	Фреймворк								
	Serenity	Cypress	Robot Framework	RedwoodHQ	Sahi	Gauge	Galen Framework	Citrus Framework	Karate-DSL
Швидкість виконання тестів	Добре	Добре	Добре	Добре	Добре	Добре	середня	Добре	середня
Зручність створення нових тестів	Добре	Добре	Добре	Добре	Добре	Добре	середня	Добре	середня

Функціональність – оцінюючи функціональність фреймворків, було важливо враховувати можливості фреймворків для виконання різних видів тестів та сценаріїв.

За результатами оцінки функціональності обраних фреймворків для автоматизованого тестування вебзастосунків, можна зробити наступні висновки:

Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge: ці фреймворки підтримують тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних та інші функціональні можливості. Вони підходять для тестування різних типів вебзастосунків.

Galen Framework: гален фреймворк спеціалізується на тестуванні складних вебзастосунків та надає засоби для перевірки макета та взаємодії з елементами інтерфейсу. Він підходить для проектів, де важливо забезпечити консистентність макету.

Citrus Framework і Karate-DSL: ці фреймворки спеціалізуються на тестуванні API та вебзастосунків. Вони надають засоби для виконання тестів взаємодії з API та можуть бути використані в проектах, де ця функціональність є ключовою.

Ключові особливості функціональних можливостей фреймворків Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Galen Framework, Citrus Framework, Karate-DSL занесені до таблиці 3.2.

Таблиця 3.2 – Порівняння ефективності тестування

Функціональні можливості	
Serenity	Підтримка тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних
Cypress	Підтримка тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних
Robot Framework	Підтримка тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних
RedwoodHQ	Підтримка тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних
Sahi	Підтримка тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних
Gauge	Підтримка тестування різних аспектів вебзастосунків, включаючи взаємодію з елементами інтерфейсу, роботу з формами, перевірку валідації даних
Galen Framework	Підтримка тестування складних вебзастосунків
Citrus Framework	Підтримка тестування API
Karate-DSL	Підтримка тестування API та вебзастосунків

Підтримка мов програмування є важливою складовою при виборі фреймворку для автоматизованого тестування (табл.3.3). Розглянувши обрані фреймворки, можна зробити такі висновки щодо їхньої підтримки мов програмування:

– Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge: ці фреймворки надають можливість працювати з різними мовами програмування. Наприклад, Serenity підтримує Java та Kotlin, Cypress використовує JavaScript, Robot Framework може бути інтегрований з різними мовами, RedwoodHQ працює зі скриптами, написаними на мові Python, Sahi підтримує Java, JavaScript та Ruby, а Gauge може бути використаний з Java, Python, Ruby та іншими мовами.

– Galen Framework: гален фреймворк використовує власний синтаксис для опису тестів та не обмежений підтримкою мов програмування. Проте він може бути інтегрований з різними мовами для автоматизації тестів.

– Citrus Framework і Karate-DSL: ці фреймворки спеціалізуються на тестуванні API та використовують власні синтаксиси для опису тестів. Citrus Framework підтримує Java, Karate-DSL підтримує Java та власний DSL.

Таблиця 3.3 – Підтримка мов програмування

Підтримка мов програмування	
Serenity	Java, Groovy, Python, C#
Cypress	JavaScript
Robot Framework	Python, Java, C#, Lua, Ruby, PHP, Perl, VBScript, Python 3, Java 8, C# 7.0
RedwoodHQ	Java, Groovy, Python, C#
Sahi	JavaScript, Java
Gauge	Java, Groovy, Python, C#

Продовження таблиці 3.3

Galen Framework	JavaScript
Citrus Framework	Java, Groovy, Python, C#
Karate-DSL	Java, Groovy, Python, C#

Також усі обрані фреймворки Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Galen Framework, Citrus Framework, Karate-DSL підтримують наступні основні браузери та операційні системи:

Підтримувані браузери:

– Google Chrome

– Mozilla Firefox

– Microsoft Edge

– Safari

– Opera

– Інші браузери

Підтримувані операційні системи:

– Windows

– macOS

– Linux

Це робить їхнє використання більш універсальним та дозволяє тестувати вебзастосунків на різних платформах і браузерах, забезпечуючи більшу покриття тестування.

Що до розширень та плагінів, всі обрані фреймворки Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Galen Framework, Citrus Framework, Karate-DSL надають різноманітні можливості розширення та підтримують плагіни для розширення їхнього функціоналу.

Розглянемо цю тему більш детально по кожному фреймворку:

– Serenity: серед популярних плагінів для Serenity є "Serenity BDD Screenplay", який дозволяє писати тести у стилі "Screenplay Pattern," і "Serenity JBehave," який додає підтримку JBehave для опису тестів;

– Cypress: Cypress дозволяє використовувати різноманітні плагіни для розширення його функціоналу, включаючи "Cypress Real Events" для реєстрації реальних подій та "Cypress Testing Library" для полегшення тестування;

– Robot Framework: Robot Framework підтримує велику кількість розширень і бібліотек, які можна легко інтегрувати для розширення функціональності та автоматизації різних видів тестів;

– RedwoodHQ: цей фреймворк дозволяє інтегрувати різноманітні плагіни, зокрема для інтеграції з іншими інструментами та системами управління проектами;

– Sahi: Sahi має власну бібліотеку плагінів, які дозволяють розширити його можливості для тестування різних аспектів вебзастосунків;

– Gauge: Gauge підтримує різноманітні плагіни для розширення його функціоналу та дозволяє інтегрувати інші інструменти;

– Galen Framework: Galen Framework має власні інструменти та плагіни для створення та виконання тестів для складних вебзастосунків;

– Citrus Framework і Karate-DSL: ці фреймворки спеціалізуються на тестуванні API і мають плагіни для розширення можливостей тестування API.

Розширення та плагіни дозволяють призначити фреймворк під конкретні потреби проекту та покращити якість автоматизованих тестів.

За результатами аналізу записаних у таблиці 3.4 різних фреймворків для автоматизації тестування, можна зробити наступні висновки щодо їхнього удобства використання та навчання:

Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, та Gauge відзначаються удобним інтерфейсом та зрозумілим синтаксисом, що робить їх дружелюбними для користувача. Це робить їх популярними виборами для

команд, які шукають фреймворк для швидкої і легкої автоматизації тестування.

Galen Framework і Citrus Framework, незважаючи на свою потужність, характеризуються складним синтаксисом і вимагають певного рівня навчання для їх використання. Вони можуть бути корисними для проектів, де необхідна велика гнучкість, але цільова команда повинна бути готова витратити час на навчання. Karate-DSL також відзначається зручним інтерфейсом та зрозумілим синтаксисом, що робить його привабливим вибором для автоматизації тестування. Цей фреймворк може бути добрим варіантом для проектів, якщо команда не має бажання витрачати багато часу на навчання.

Таблиця 3.4 – Зручність використання та навчання

Назва	**Інтерфейс**	**Синтаксис**
Serenity	+	+
Cypress	+	+
Robot Framework	+	+
RedwoodHQ	+	+
Sahi	+	+
Gauge	+	+
Galen Framework	+	-
Citrus Framework	+	-
Karate-DSL	+	+

Аналіз підтримки спільноти та доступності документації для обраних фреймворків для автоматизованого тестування вебзастосунків записанно у таблиці 3.5.

Таблиця 3.5 – Аналіз підтримки спільноти та доступності

	Спільнота	**Документація**
Serenity	Фреймворк Serenity має активну та досить численну спільноту користувачів, яка активно обговорює питання, надає поради та підтримку один одному. Це свідчить про великий інтерес та популярність Serenity серед тестерів.	Документація для Serenity є детальною, структурованою та легко доступною. Вона включає приклади, пояснення та рекомендації, які допомагають користувачам оволодіти фреймворком швидко та ефективно.
Cypress	має активну та широку спільноту користувачів, яка завжди готова надавати допомогу та ділитися знаннями. Це свідчить про велику популярність фреймворку в спільноті тестерів.	Документація Cypress є детальною та зрозумілою. Включає приклади, рекомендації та опис функціональності, що допомагає користувачам швидко ознайомитися з фреймворком.
Robot Framework	Має велику кількість активних користувачів та розробників, готових надавати підтримку та допомогу.	Дуже добре розроблена і містить вичерпні пояснення, приклади та рекомендації для користувачів. Вона створена з урахуванням потреб різних категорій користувачів..

Продовження таблиці 3.5

RedwoodHQ	Спільнота RedwoodHQ менша порівняно з іншими фреймворками, але вона все ж активно взаємодіє та надає підтримку користувачам.	Документація для RedwoodHQ існує, але може бути менш детальною порівняно з іншими фреймворками, що може ускладнити оволодіння ним.
Sahi	Спільнота користувачів Sahi менша, але вона надає підтримку та допомогу один одному.	Документація Sahi існує, але може вимагати покращення для забезпечення більшої зрозумілості та доступності.
Gauge	Спільнота Gauge зростає, і вона надає підтримку користувачам. Це свідчить про потенціал фреймворку у майбутньому.	Документація Gauge є, але може вимагати додаткового розширення та покращення для забезпечення більшої зрозумілості.

Продовження таблиці 3.5

Galen Framework	Спільнота користувачів Galen Framework менша порівняно з іншими фреймворками.	Документація Galen Framework існує, але синтаксис фреймворку вимагає особливого навчання, тож додаткова документація може бути корисною.
Citrus Framework	Спільнота користувачів Citrus Framework менша, але існує активний обмін інформацією та підтримкою.	Документація Citrus Framework може бути менш розгорнутою, але присутність спільноти допомагає вирішувати питання.
Karate-DSL	Фреймворк Karate-DSL має активну та ростущу спільноту користувачів, яка надає підтримку та ділиться досвідом.	Документація Karate-DSL є доступною та детальною, що допомагає користувачам швидко освоювати фреймворк.

На підставі аналізу підтримки спільноти та доступності документації можна визначити, що Serenity, Cypress, та Robot Framework мають активні та великі спільноти користувачів і високоякісну документацію. Karate-DSL також відзначається активною спільнотою та якісною документацією. Водночас, Galen Framework та Citrus Framework мають менші спільноти та докладають зусиль для покращення документації. RedwoodHQ, Sahi, Gauge мають меншу популярність, але все ж мають активну спільноту і документацію.

3.3 Аналіз результатів дослідження

Дослідивши та порівнявши фреймворки для автоматизованого тестування вебзастосунків, змогли отримати важливі уявлення про їхні особливості та можливості.

Перш за все, аналіз показав, що існує безліч фреймворків, призначених для автоматизованого тестування вебзастосунків, і кожен з них має свої переваги та обмеження.

Serenity – це фреймворк з широкими функціональними можливостями та гарною підтримкою різних браузерів та платформ. Він також має гарну документацію та підтримку спільноти. Однак він може бути складним у вивченні та використанні.

Cypress – це фреймворк з відмінною продуктивністю та зручним API. Він також має гарну підтримку різних браузерів та платформ. Однак він може бути дорогим для використання у великих проектах.

Robot Framework – це фреймворк із широкими функціональними можливостями та гарною підтримкою різних мов програмування. Він також

має гарну документацію та підтримку спільноти. Однак він може бути менш продуктивним, ніж інші фреймворки.

RedwoodHQ – це фреймворк із широкими функціональними можливостями та гарною підтримкою різних браузерів та платформ. Він також має гарну документацію та підтримку спільноти. Однак він може бути менш продуктивним, ніж інші фреймворки.

Sahi – це фреймворк із широкими функціональними можливостями та гарною підтримкою різних браузерів та платформ. Він також має гарну документацію та підтримку спільноти. Однак він може бути менш продуктивним, ніж інші фреймворки.

Gauge – це фреймворк із широкими функціональними можливостями та гарною підтримкою різних мов програмування. Він також має гарну документацію та підтримку спільноти. Однак він може бути менш продуктивним, ніж інші фреймворки.

Galen Framework – це фреймворк із вузькою спеціалізацією на тестуванні інтерфейсу користувача. Він має гарну документацію та підтримку спільноти. Однак він може бути менш гибким, ніж інші фреймворки.

Citrus Framework – це фреймворк з широкими функціональними можливостями та гарною підтримкою різних мов програмування. Він також має гарну документацію та підтримку спільноти. Однак він може бути менш продуктивним, ніж інші фреймворки.

Karate-DSL – це фреймворк із простим і зрозумілим API. Він також має гарну документацію та підтримку спільноти. Однак він може бути менш гнучким, ніж інші фреймворки. Для проектів, де важлива швидка імплементація, він може бути ефективним рішенням.

Для проектів з високими вимогами до продуктивності та гнучкості: Cypress, Karate-DSL (рис 3.1).

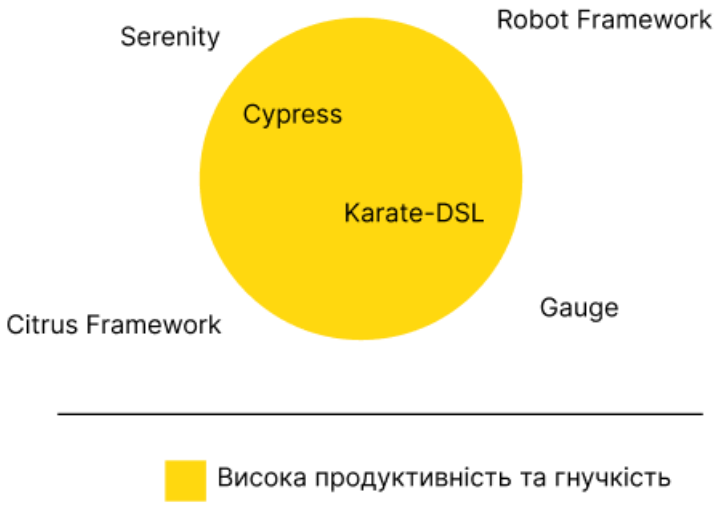

Рисунок 3.1 – Діаграма фреймворків продуктивності та гнучкості

Cypress – це чудовий вибір для проектів, що потребують високої продуктивності та гнучкості. Він має зручний API, що дозволяє розробникам створювати швидкі та ефективні тести.

Karate-DSL – це фреймворк із простим і зрозумілим API. Він може бути хорошим вибором для проектів, які потребують швидкого впровадження.

Для проектів із широкими функціональними можливостями: Serenity, Robot Framework, Gauge, Citrus Framework (рис.3.2).

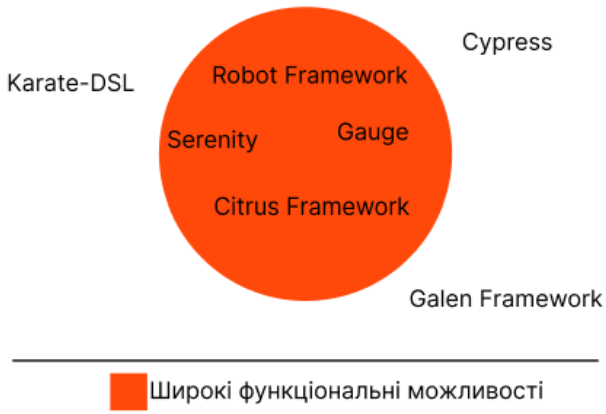

Рисунок 3.2 – Діаграма фреймворків широкі функціональні можливості

Serenity – це фреймворк із широким набором функцій, включаючи підтримку різних браузерів та платформ, інтеграцію з Jira та Confluence, а також звіти за результатами тестування.

Robot Framework – це фреймворк із широким набором функцій, включаючи підтримку різних мов програмування, інтеграцію з різними інструментами та звіти за результатами тестування.

Gauge – це фреймворк із широким набором функцій, включаючи підтримку різних мов програмування, інтеграцію з різними інструментами та звіти за результатами тестування.

Citrus Framework – це фреймворк із широким набором функцій, включаючи підтримку різних мов програмування, інтеграцію з різними інструментами та звіти за результатами тестування.

Для проектів з обмеженим бюджетом: Robot Framework (рис. 3.3).

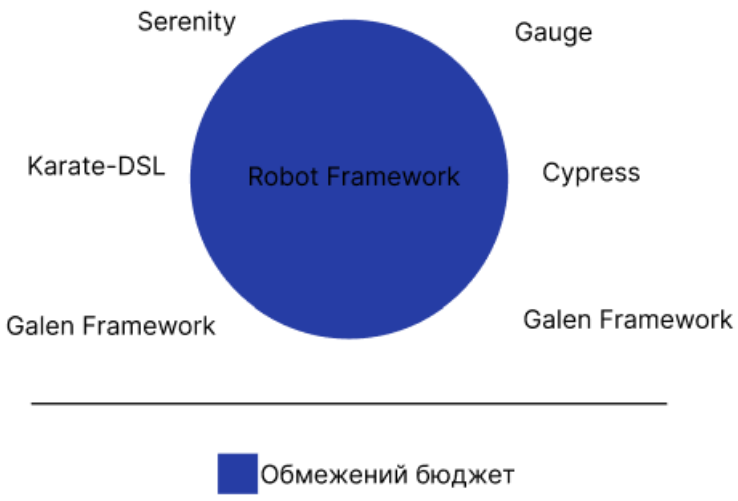

Рисунок 3.3 – Діаграма фреймворків з обмеженим бюджетом

Robot Framework – це безкоштовний та відкритий фреймворк, який може бути гарним вибором для проектів з обмеженим бюджетом.

Для проектів із вузькою спеціалізацією: Galen Framework (рис 3.4).

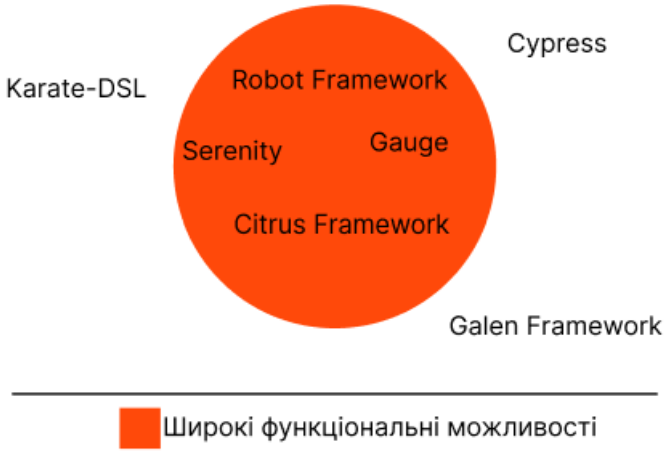

Рисунок 3.3 – Діаграмма фреймворків із вузькою спеціалізацією.

Galen Framework – це фреймворк з вузькою спеціалізацією на тестуванні інтерфейсу користувача. Він може бути хорошим вибором для проектів, в яких потрібне автоматизоване тестування інтерфейсу користувача.

Результати у вигляді схематичного зображенні диаграми Венна можно побачити на рисунку 3.4.

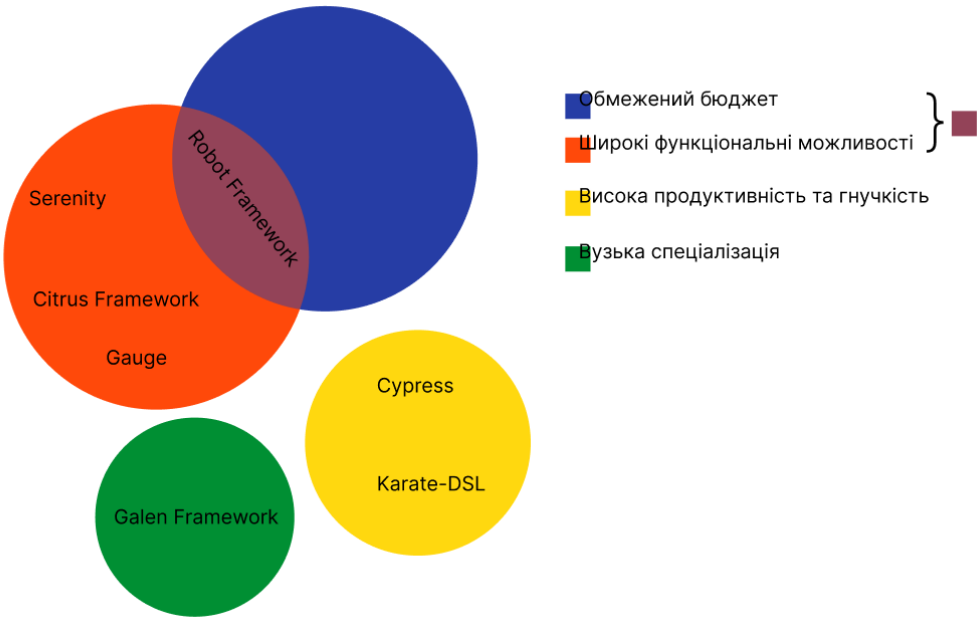

Рисунок 3.4 – Схематичне зображення фреймворків.

ВИСНОВКИ

У цій кваліфікаційної роботі вроведено порівняльний аналіз різних фреймворкіх для автоматизованого тестування вебзастосунків. Робота була спрямована на визначення переваг та обмежень кожного з розглянутих фреймворків, а також на розробку рекомендацій для вибору найкращого фреймворку в залежності від вимог і особливостей конкретного проекту.

Проведено огляд літератури, вивчено основні поняття та терміни в галузі автоматизованого тестування, досліджено історію розвитку фреймворків для автоматизованого тестування та попередні дослідження та порівняння фреймворків. В результаті цього огляду, набуто глибокого розуміння та експертної належності до предметної області.

Також проведено докладний аналіз різних фреймворків, включаючи Serenity, Cypress, Robot Framework, RedwoodHQ, Sahi, Gauge, Citrus Framework, Karate-DSL і Galen Framework. Визначено основні переваги та недоліки кожного з них, а також визначено, які типи проектів найкраще підходять для використання кожного з фреймворків.

У підсумку, кваліфікаційна робота стала цінним джерелом інформації для тих, хто шукає найкращий фреймворк для автоматизованого тестування вебзастосунків. Ретельно дослідили та порівняли різні фреймворки, надали докладний аналіз їхніх переваг і недоліків, і надали рекомендації щодо вибору фреймворку відповідно до конкретних потреб і вимог проекту.

Ця робота покликана сприяти розвитку сфери автоматизованого тестування та допомогти у виборі оптимальних інструментів для їхніх завдань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Каратанов, О., Єна, М., Бова, Є., & Устименко, О. (2021). Порівняння популярних тестових фреймворків JUnit і TestNG. Молодий вчений, (5), 164–170. URL: http://dx.doi.org/10.32839/2304-5809/2021-5-93-31. (дата звернення: 17.11.2023)

2. Таран, Е. (2023). Досвід тестувальника Playwright vs Selenium. Форум "ДОУ". URL: https://dou.ua/forums/topic/45780/ (дата звернення: 15.11.2023)

3. Serenity/JS. (2023). Getting started | Serenity/JS. URL: https://serenity-js.org/handbook/thinking-in-serenity-js/index.html (дата звернення: 15.10.2023)

4. India vs Australia 3rd T20I Highlights: India beat Australia by 6 … (2023). The Times of India. URL: https://rb.gy/yqlwy5 (дата звернення: 15.10.2023)

5. IND vs AUS 3rd ODI, Highlights: Australia win series 2-1, become No.1 (2023). Indian Express. URL: https://rb.gy/3bc6qt (дата звернення: 15.10.2023)

6. Serenity/JS. (2023). Serenity/JS. URL: https://serenity-js.org/ (дата звернення: 16.10.2023)

7. Cypress Web Testing Framework: Getting Started | BrowserStack. (2023). BrowserStack. URL: https://www.browserstack.com/guide/cypress-framework-tutorial (дата звернення: 25.10.2023)

8. A Step-By-Step Guide To Cypress API Testing | LambdaTest. (2023). LambdaTest. URL: https://www.lambdatest.com/blog/cypress-api-testing/ (дата звернення: 26.10.2023)

9. Robot Framework. (2023). Robot Framework. URL: https://robotframework.org/ (дата звернення: 01.11.2023)

10. Robot Framework User Guide. (2023). Robot Framework User Guide. URL: https://u.to/CVUlIA (дата звернення: 12.10.2023)

11. Welcome to | ROBOT FRAMEWORK. (2023). Welcome to | ROBOT FRAMEWORK. URL: https://robotframework.org/ (дата звернення: 14.10.2023)

12. RedwoodHQ | Open Source Test Automation Framework. (2023). RedwoodHQ | Open Source Test Automation Framework. URL: http://redwoodhq.com/ (дата звернення: 10.10.2023)

13. GitHub - dmolchanenko/RedwoodHQ. (2023). GitHub - dmolchanenko/RedwoodHQ. URL: https://github.com/dmolchanenko/RedwoodHQ (дата звернення: 22.10.2023)

14. India vs Australia 3rd ODI, Highlights: Australia win series 2-1, become No.1 (2023). Indian Express. URL: https://u.to/OVUlIA (дата звернення: 28.10.2023)

15. Sahi Framework - Sahi Pro. (2023). Sahi Framework - Sahi Pro. URL: https://sahipro.com/sahi-framework/ (дата звернення: 29.10.2023)

16. BDTA | Business Driven Test Automation | Sahi Pro. (2023). BDTA | Business Driven Test Automation | Sahi Pro. URL: https://sahipro.com/business-driven-test-automation/ (дата звернення: 09.10.2023)

17. GitHub - obss/sahi: Framework agnostic sliced/tiled inference …. (2023). GitHub - obss/sahi: Framework agnostic sliced/tiled inference …. URL: https://github.com/obss/sahi (дата звернення: 11.10.2023)

18. Test Automation Using the Gauge Framework. (2023). Guru99. URL: https://www.guru99.com/gauge-test-automation.html (дата звернення: 27.10.2023)

19. Open Source Test Automation Framework | Gauge. (2023). Gauge. URL: https://gauge.org/ (дата звернення: 08.10.2023)

20. Reducing the complexity of test automation using Gauge. (2023). InfoWorld. URL: https://www.infoworld.com/article/3534133/reducing-the-complexity-of-test-automation-using-gauge.html (дата звернення: 27.10.2023)

21. Citrus Framework. (2023). Citrus Framework. URL: https://citrusframework.org/ (дата звернення: 05.10.2023)

22. Citrus Framework. (2023). Citrus Framework. URL: https://citrusframework.org/docs/overview/ (дата звернення: 21.10.2023)

23. GitHub - citrusframework/citrus: Framework for automated integration (2023).URL: https://github.com/citrusframework/citrus (дата звернення: 22.10.2023)

24. Citrus API Automation Testing Framework. (2023). ToolsQA. URL: https://www.toolsqa.com/citrus-framework/ (дата звернення: 01.10.2023)

25. Galen Framework | Automated testing of responsive design. (2023). Galen Framework. URL: http://galenframework.com/ (дата звернення: 27.10.2023)

26. Documentation | Galen Framework. (2023). Documentation | Galen Framework. URL: http://galenframework.com/docs/getting-started-install-galen/ (дата звернення: 09.10.2023)

27. Galen Framework - Wikipedia. (2023). Galen Framework - Wikipedia. URL: https://en.wikipedia.org/wiki/Galen_Framework(датазвернення:26.10.2023)

28. Responsive Design Testing with Galen Framework. (2023). Guru99. URL: https://www.guru99.com/responsive-design-testing-galen-framework.html (дата звернення: 15.10.2023)

29. Getting Started On Browser Automation With Galen Framework - LambdaTest. (2023). LambdaTest. URL: https://www.lambdatest.com/blog/getting-started-on-browser-automation-with-galen-framework/(датазвернення:15.10.2023)

30. Krzyzanowski, W. (2023). Karate DSL — Automatizando testes de API de forma simples. [Online]. URL: https://medium.com/@wkrzyzanowski/karate-dsl-automatizando-testes-de-api-de-forma-simples-1d44322b4793 (дата звернення: 01.10.2023)

31. Step-by-Step: Testing APIs with KarateDSL – Sweetcode.io. (2023). Sweetcode.io. URL: https://sweetcode.io/step-step-testing-apis-karatedsl (дата звернення: 14.10.2023)

32. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

33. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

34. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

35. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

36. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

37. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

38. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.