Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Автоматизація тестування системи керування навчальним процесом JetIQ

Виконав: студент 2 курсу, групи
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Андрій СИМОН
Ім'я ПРІЗВИЩЕ

Керівник: д.т.н., професор, професор каф. КСУ
степінь, звання, посада

Володимир ДУБОВОЙ
Ім'я ПРІЗВИЩЕ

«01» 12 2023 р.

Опонент: к.т.н., доцент, професор каф. АІІТ
степінь, звання, посада

Євген ПАЛАМАРЧУК
Ім'я ПРІЗВИЩЕ

«05» 12 2023 р.

Допущено до захисту

Т.в.о.Зав. кафедри КСУ

_Марія ЮХИМЧУК

«07» 12 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра комп'ютерних систем управління

Рівень вищої освіти другий (магістерський)

Галузь знань – 15 – Автоматизація та приладобудування

Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології

Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Симону Андрію Дмитровичу.
(прізвище, ім'я, по батькові)

1. Тема роботи: Автоматизація тестування системи керування навчальним процесом JetIQ
керівник роботи д.т.н., професор, професор кафедри КСУ Дубовой В. М.
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи " 1 " грудня 2023 року
3. Вихідні дані до роботи: тестовий фреймворк, автоматизація тестування системи, CI/CD, результати тестування, логи, скріншоти, Bandana O. M., 2018. Selenium Testing Interview Q&A. Kyiv.
4. Зміст текстової частини: вступ, аналіз галузі автоматизованого тестування, аналіз системи керування навчальним процесом JetIQ та задач її тестування, розробка фреймворку для автоматизованого тестування, тестування та експериментальне застосування фреймворку, аналіз економічних показників розробки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): титульний слайд; актуальність теми; мета, об'єкт, предмет та задачі дослідження; новизна та практична цінність одержаних результатів; Selenium WebDriver; вимоги, які підлягають тестуванню; список тест-кейсів; матриця трасування; архітектура фреймворку; алгоритм роботи тестового сценарію; блок-схема CI/CD пайплайну; застосування фреймворку: тестування системи локально; застосування фреймворку: тестування системи на CI/CD; економічна частина; апробація та публікація результатів роботи; фінальний слайд.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
5	професор, Бурєннікова Н. В.	09.10.2023	01.11.2023

2. Дата видачі завдання "09" жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз галузі автоматизованого тестування	09.10.2023	16.10.2023	
2	Розробка тестових сценаріїв, матриці трасування та архітектури фреймворку	17.10.2023	21.10.2023	
3	Розробка алгоритму роботи тестового сценарію та CI/CD схеми фреймворку	22.10.2023	25.10.2023	
4	Розробка програмних компонент для автоматизованого тестового фреймворку	26.10.2023	16.11.2023	
5	Тестування та експериментальне застосування фреймворку	17.11.2023	19.11.2023	
6	Розрахунок економічної частини	20.11.2023	27.11.2023	
7	Графічні матеріали	28.11.2023	30.11.2023	

Студент (підпис) Андрій СИМОН (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи (підпис) Володимир ДУБОВОЙ (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 621.374.415

Симон А. Д. Автоматизація тестування системи керування навчальним процесом JetIQ. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2023. 157 с.

На укр. мові. Бібліогр.: 52; рис.: 43; табл. 21.

У магістерській кваліфікаційній роботі розроблено фреймворк для автоматизації тестування системи керування навчальним процесом JetIQ. У оглядово-аналітичній частині роботи розглянуто особливості автоматизації тестування, а також охарактеризовано освнові підходи та методи до розробки тестових фреймвоків, обґрунтована доцільність роботи. У теоретично-методичній частині розроблено моделі бізнес-процесу системи керування навчальним процесом JetIQ, обгрунтовано інструментальні засоби реалізації системи. Також розроблено ряд тестових сценаріїв та запропоновано архітектуру тестового фреймворку. У практичній частині розроблено програмне забезпечення, наведено результати його роботи. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки. Ілюстративна частина складається з 16 плакатів із результатами роботи.

Ключові слова: автоматизація, тестування, розробка через поведінку, тестовий фреймворк.

ANNOTATION

UDC 621.374.415

Symon A. D. Automation of testing of the educational process management system JetIQ. Master's qualification work in specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2023. 157 c.

In Ukrainian. Bibliography: 52; Figures: 43; Table 21.

In the master's qualification work, a framework for automating the testing of the JetIQ learning management system was developed. In the review and analytical part of the work, the features of test automation are considered, as well as the main approaches and methods for developing test frameworks are characterized, and the feasibility of the work is substantiated. In the theoretical and methodological part, the business process models of the JetIQ learning management system are developed, and the tools for implementing the system are substantiated. A number of test scenarios have been developed and the architecture of the test framework has been proposed. In the practical part, the software is developed and the results of its operation are presented. In the economic part, the technical level is analyzed and the cost of development is calculated. The illustrative part consists of 16 posters with the results of the work.

Keywords: automation, testing, behavior-driven development, test framework.

ЗМІСТ

ВСТУП 8
1. АНАЛІЗ ЗАДАЧІ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ СИСТЕМИ КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ 11
1.1 Аналіз технологій автоматизації тестування 11
1.2 Аналіз існуючих підходів для розробки тестових фреймворків 16
1.3 Аналіз існуючих методологій розробки 24
1.4 Обґрунтування вибору мови програмування та середовища розробки 28
1.5 Обґрунтування задач МКР 34
1.6 Обґрунтування новизни роботи 36
1.7 Висновки 38
2. АНАЛІЗ СИСТЕМИ КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ JETIQ ТА ЗАДАЧ ЇЇ ТЕСТУВАННЯ 39
2.1 Аналіз життєвого циклу та моделі розробки системи 39
2.2 Аналіз вимог до системи, розробка тест-кейсів та матриці трасування 44
2.3 Розробка архітектури фреймворку 52
2.4 Розробка алгоритму роботи автоматизованого тестового сценарію 56
2.5 Розробка CI/CD пайплайну 58
2.6 Висновки 62
3. РОЗРОБКА ФРЕЙМВОРКУ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ 63
3.1 Реалізація тестових сценаріїв за допомогою BDD підходу 63
3.2 Програмна реалізація фреймворку. 65
3.3 Налаштування CI/CD пайплайну 69
3.4 Висновки 73
4. ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ЗАСТОСУВАННЯ ФРЕЙМВОРКУ 74
4.1 Інструкція користувача 74
4.2 Проведення локального тестування. 77
4.3 Проведення тестування за допомогою CI/CD пайплайну 79
4.4 Висновки. 81
5. АНАЛІЗ ЕКОНОМІЧНИХ ПОКАЗНИКІВ РОЗРОБКИ 82

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки 82
5.2 Розрахунок узагальненого коефіцієнта якості розробки. 86
5.3 Розрахунок витрат на проведення науково-дослідної роботи..... 89
5.4 Розрахунок економічної ефективності науково-технічної розробки від її впровадження..... 99
5.5 Висновки. 105
ВИСНОВКИ..... 107
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 108
ДОДАТКИ..... 113
Додаток А. Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень 114
Додаток Б. Технічне завдання..... 115
Додаток В. Лістинг програми 119
Додаток Г. Ілюстративна частина..... 149

ВСТУП

Актуальність теми. Сьогодні розробка продукту без якісного та ретельного тестування є неможливою. Тестування — це метод технічного дослідження, призначений для отримання інформації про якість продукту відповідно до ситуації, в якій він має бути використаний.

Основним методом тестування є оцінка та максимізація значення кожного етапу життєвого циклу розроблення програмного забезпечення, щоб досягти необхідних стандартів якості, продуктивності та доступності. Таким чином, розробники приділяють велику увагу перевірці працездатності веб-додатків у глобальному та локальному середовищах. Раніше все це виконувалося вручну. Десятки людей тестували сотні сценаріїв у всіх браузерах, виявляючи проблеми та намагаючись з’ясувати, чому вони виникають. Такий вид роботи вимагав багато кваліфікованих працівників та часу, що негативно впливає на бізнес.

Ця проблема особливо актуальна для інтеграційного тестування, стадії тестування програмного забезпечення, під час якої окремі модулі програми поєднуються та тестуються разом. Інтеграційне тестування проводиться з метою підтвердження того, що основні частини програми відповідають вимогам щодо функціональності, продуктивності та надійності.

Таким чином, необхідно автоматизувати тестові сценарії. Використання спеціальних програмних інструментів для виконання тестових сценаріїв і перевірки функціональності програмного забезпечення без прямого втручання людей називається автоматизацією тестування. Автоматизація тестування має на меті полегшити та пришвидшити процес тестування, а також підвищити його ефективність і надійність.

Переваги автоматизації тестування включають у собі значний економічний виграш, зменшення часу на виконання тестів, збільшення покриття тестами та зниження ймовірності помилок людського фактору. Автоматизовані тести можуть

бути виконані швидше та ефективніше, що особливо важливо в умовах швидких ітерацій розробки та випуску нових версій програмного забезпечення.

Автоматизація тестування є дуже актуальною темою в сучасному програмній розробці з ряду причин:

1) Ефективність та повторюваність: автоматизовані тести можуть бути виконані швидше і більш ефективно, а також повторювані безперервно при кожній зміні коду чи новій збірці. Економія ресурсів: хоча витрати на створення автоматизованих тестів можуть бути високими на початку, з часом вони економлять час та ресурси, які розробники та тестувальники витрачають на ручне тестування.

2) Виявлення помилок на ранніх етапах: автоматизоване тестування дозволяє виявляти помилки на ранніх етапах розробки, коли виправлення коштує менше.

3) Покращення якості продукту: автоматизовані тести можуть забезпечити широкий охоплення тестування, допомагаючи виявляти помилки та недоліки, які можуть залишитися непоміченими при ручному тестуванні.

4) Неперервна інтеграція та постійна поставка: автоматизація тестів є ключовою частиною неперервної інтеграції та постійної поставки (CI/CD), що дозволяє швидко та безпечно впроваджувати новий функціонал.

Враховуючи ці переваги, багато компаній активно впроваджують практики автоматизації тестування в своїх проектах для покращення продуктивності, якості та швидкості впровадження нових функцій.

Тому розробка фреймворку, який є зрозумілим і простим у використанні та дозволяє автоматизувати тестування, є досить актуальною задачею.

Мета і завдання дослідження. Метою даної роботи є підвищення ефективності системи підтримки навчального процесу JetIQ шляхом автоматизації її тестування.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

- аналіз задачі автоматизації тестування системи керування навчальним процесом;

- аналіз системи керування навчальним процесом JetIQ та задач її тестування;
- розробка фреймворка для автоматизованого тестування;
- тестування та експериментальне застосування фреймворку;
- аналіз економічних показників розробки.

Об’єктом дослідження є процес автоматизація процесу тестування.

Предметом дослідження є методи та засоби для автоматизація тестування для системи керування навчальним процесом JetIQ.

Новизна отриманих результатів.

1) запропоновано оновлений підхід розробки фреймворку для автоматизації тестування, який поєднує в собі кілька технологій (JUnit, Maven, Selenium, Cucumber, CI/CD). Це дало можливість підвищити швидкість та якість тестування, спростити написання та підтримку нових автоматизованих тестових сценаріїв;

2) удосконалено підхід до розробки тестових сценаріїв, що на відміну від базового, використовує можливості використаних технологій. Це дало можливість писати тести зрозумілою звичайною мовою (наприклад, англійською), запускати тести на віддаленій машині та вдвічі скоротити час проходження тестів.

Практичною цінністю є алгоритм та розроблені програмні засоби для тестового фреймворку, за допомогою якого можна автоматизувати тестування системи керування навчальним процесом JetIQ.

Апробація. Представлені в роботі результати апробовані в результаті участі в конференції Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)».

Публікації: Симон А. Д., Дубовой В. М. «Використання технології WebDriver для автоматизації тестування», «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19645.

1. АНАЛІЗ ЗАДАЧІ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ СИСТЕМИ КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

1.1 Аналіз технологій автоматизації тестування

Коли є обмежені тимчасові ресурси та високі вимоги до якості програмних продуктів, помилки та дефекти — це нормальне явище в процесі створення програмних продуктів. Крім того, необхідно створити ефективну систему управління та контролю за всіма етапами тестування [1]. Сьогодні неможливо автоматизувати кожен етап тестування, що робить неможливим контроль якості програмного забезпечення [2].

Ручне тестування є трудомістким, дорогим і часто монотонним процесом. Це викликає труднощі, особливо коли є обмежені ресурси та жорсткі терміни [3]. Важливо перейти до автоматизації всіх ручних завдань тестування, якщо вам потрібно покращити тестування додатків, щоб переконатися, що вони працюють правильно.

Автоматизація тестування дозволяє оптимізувати якість складних програмних продуктів ефективним по вартості способом за прийнятний час тому, що інструментальні засоби автоматизації записують взаємодію користувачів з програмним продуктом, а сформовані на цій основі сценарії використовуються для подальших тестів [4]. Це збільшує швидкість виробництва більш високоякісного програмного забезпечення.

Процес автоматизації тестування складається з трьох етапів [5]:

– Запис. Для перевірки відповіді системи сценарій тестування може включати точки верифікації. Крім того, сценарії тестування можуть бути залежними від тестових даних, щоб виконувати один і той самий скрипт з абсолютно різними наборами вхідних даних.

– Поліпшення шляхом додавання коду, який має кілька функцій. Умовне галуження, рефакторинг і обробка виняткових ситуацій є типовими змінами сценаріїв тестування.

– Відтворення. Виконання сценаріїв, які імітують дії користувача додатку під час запису тесту. Регресійне тестування фіксує розбіжності, що дозволяє тестувальникам визначити, чи добре працює система та чи було виявилено які-небудь проблеми.

Операційні системи DOS і CP/M були часом появи «автоматизації» [6]. Тоді вона працювала над додатком команд через командний рядок і перевіряла результати. Після цього додалися віддалені виклики за допомогою API для роботи з мережею [7]. Однак автоматизація тестування почала реалізуватися лише в 1980-х роках [8].

Тестування на рівні коду та тестування GUI є двома основними методами автоматизації тестування [9]. Першого типу належить модульне тестування. До другого етапу спеціальні тестові фреймворки використовуються для моделювання дій користувача.

Найпоширенішою формою автоматизації є тестування програм за допомогою GUI [10]. Популярність цього типу тестування пояснюється двома причинами. По-перше, програму можна тестувати таким же чином, як його використовують люди, і по-друге, можна тестувати програму, не маючи доступу до вихідного коду [11].

Трудомісткість є основною проблемою автоматизованого. Незважаючи на те, що автоматизація дозволяє усунути деякі рутинні операції та прискорити виконання тестів, оновлення самих тестів може вимагати значних ресурсів [12]. Це стосується кожного з двох типів автоматизації. Також слід зазначити, що при змінені інтерфейсу програми треба знову переписувати усі тести, котрі пов'язані з оновленими вікнами, що може відняти значні ресурси при великій загальній кількості тестів [13].

Автоматизовані тестові фреймворки є вирішальним компонентом автоматизації тестування. Автоматизовані тестові фреймворки — це програми та середовища, які допомагають автоматизувати виконання тестів програмного забезпечення. Розробники можуть використовувати різні системи та технології для автоматизації тестування. Нижче наведено деякі з найбільш поширених тестових фреймворків і аналізи їх:

1) Selenium WebDriver. Selenium — це набір інструментів, призначених для створення веб-додатків. WebDriver - це частина Selenium, яка надає API для взаємодії з веб-елементами. На рис. 1.1 показано архітектуру Selenium WebDriver.

Переваги:

+ Підтримка різних браузерів;

+ Підтримка багатьох мов програмування (Java, Python, C#, і т. д.).

+ Широке використання в індустрії.

Недоліки:

– Може вимагати значних ускладнень для роботи з деякими елементами і AJAX.

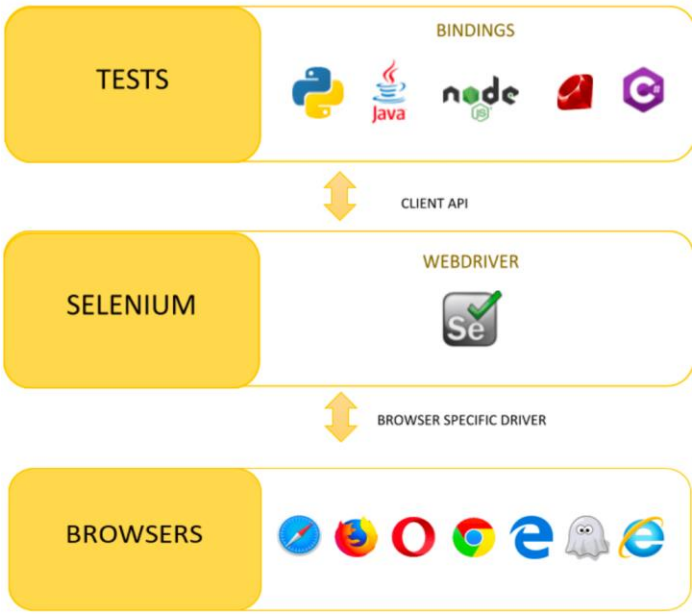


Рисунок 1.1 – Архітектура Selenium WebDriver

2) Appium — фреймворк для автоматизації мобільних додатків. На рис. 1.2 показано архітектуру Appium.

Переваги:

+ Підтримка різних мов програмування.

+ Відкритий код.

+ Кросплатформеність: може взаємодіяти з додатками для різних платформ.

Недоліки:

– Деяка складність налаштування середовища для мобільних тестів.

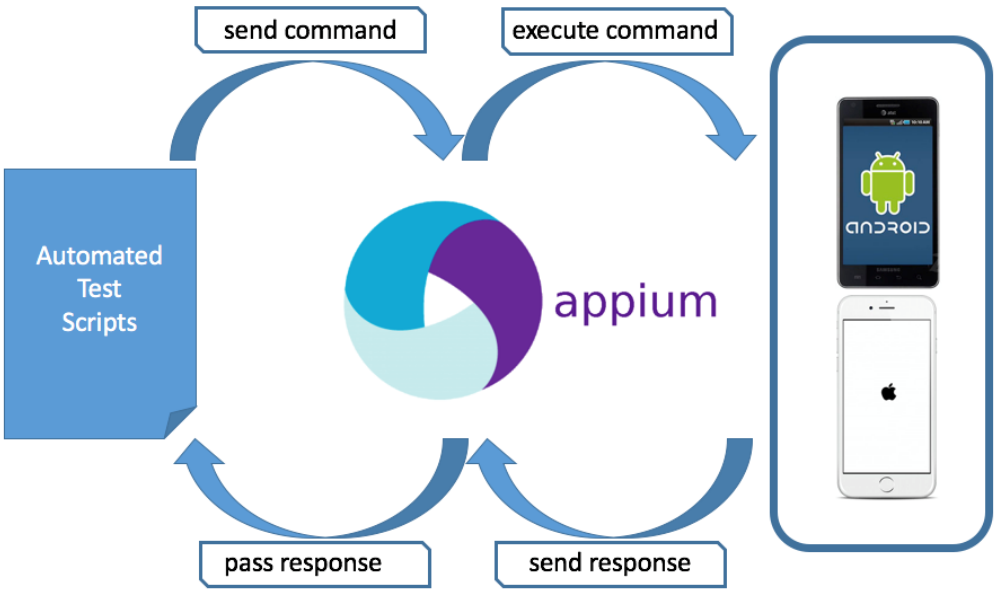


Рисунок 1.2 – Архітектура Appium

3) JUnit та TestNG — фреймворки для тестування Java-програм. На рис. 1.3 показано архітектуру JUnit.

Переваги:

+ Велика спільнота та підтримка.

+ Простота використання та інтеграція з іншими інструментами.

Недоліки:

– Специфічні для Java, що є обмежуючим для деяких проектів.

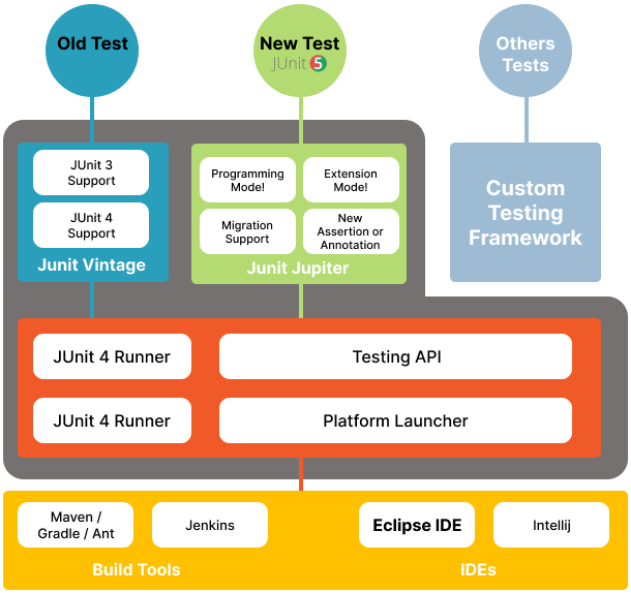


Рисунок 1.3 – Архітектура JUnit

4) Cypress — фреймворк для тестування веб-додатків, який працює в браузері. На рис. 1.4 показано архітектуру Cypress .

Переваги:

+ Потужна система виявлення помилок.

+ Легкість налаштування та використання.

+ Зручний для тестування SPA (односторінкових) додатків.

Недоліки:

– Обмежена підтримка браузерів порівняно з Selenium.

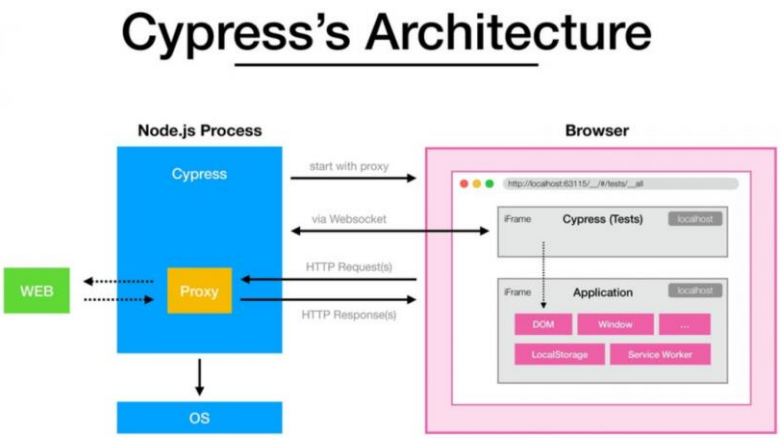


Рисунок 1.4 – Архітектура Cypress

Нарешті, автоматизовані тестові системи покращують якість програмного забезпечення, прискорюють процес розробки та зменшують ризики, пов’язані з випуском нових версій програмного забезпечення.

Збільшення ефективності, охоплення та прискорення тестування під час постійного повтору тестових сценаріїв є основними завданнями впровадження автоматизації в процес тестування [14]. Автотест можна запускати на регулярній основі як під час робочого, так і поза робочим часом. Пошук і запис помилок у ручних тестах займає в середньому один день [15]. Автоматизація процесу скоротить час і дозволить виявити помилки в коді, коли він вноситься в репозиторій вихідного коду [16].

1.2 Аналіз існуючих підходів для розробки тестових фреймворків

Тестування – це необхідний етап при розробці будь-якого програмного продукту. Автоматизація допомагає прискорити й полегшити даний процес. Проте слід зазначити, що не усі види та типи тестування потребують автоматизації, а лише ті й тільки ті, що засновані на діях, які повторюються [17].

Вибір потрібного інструменту безперечно залежить від того, які вимоги ставляться перед тестовими сценаріями. У більшості випадків використовуються кілька інструментів одночасно, кожен з яких оцінює свою ступінь системної архітектури [18].

Завдяки використанню інструментів автоматизації тестування можна записувати та реєструвати відтворені тести, а також повторно запускати їх у різні моменти часу [19].

Наявність таких функцій є важливою при їх виборі:

– Інтуїтивно зрозумілий інтерфейс та простота у використанні

– Формування інформативних звітів про тестування

– Вбудована підтримка основних типів тестування.

Існує перелік ефективних і рекомендованих методів тестування [20]. Частина з них показана на рис. 1.5. Вони дозволяють автоматизувати процеси, пов’язані з різними продуктами.

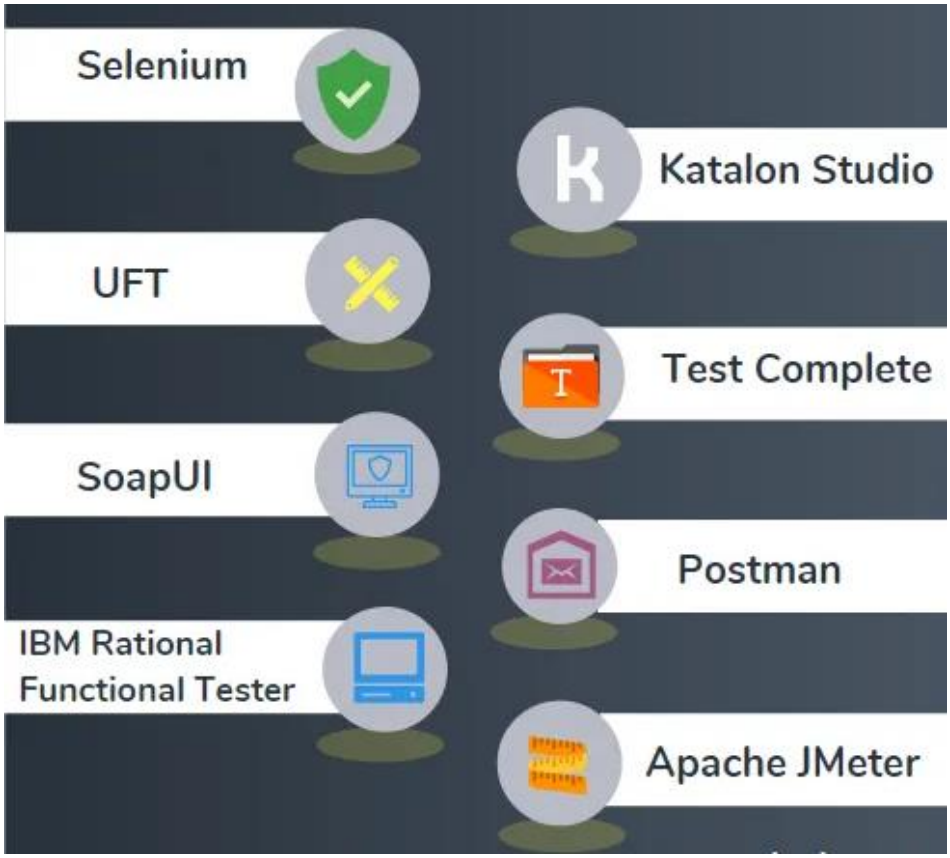


Рисунок 1.5 – Приклад інструментів для автоматизації тестування

Розглянемо основні інструменти автоматизації тестування [21]:

– Selenium
– HPE UFT
– TestingWhiz
– Postman
– JMeter

Selenium може тестувати веб-сайти та програми в різних браузерах і операційних системах [22]. Його перевага полягає в тому, що тестувальники можуть вибрати мову програмування, яку вони використовують для написання

своїх тестів. Застосунок запису/відтворення Selenium дозволяє тестувати веб-застосунки, не знаючи мов програмування. Крім того, інструмент підтримує Selenese, предметно-орієнтовану мову, яка використовується для написання тестів на таких мовах, як C#, Java, Python, Perl, Groovy, Ruby, PHP та Scala [23]. Найпоширенішим завданням є автоматизація Web-тестування. Коли люди говорять про Selenium, вони зазвичай мають на увазі Selenium WebDriver [24]. Основна частина процесу розробки продукту зосереджена саме на цій частині. Автоматизація тестування WebDriver часто порівнюється з роботою водія таксі. У тестуванні та водінні таксі беруть участь три елементи: замовник (інженер-випробувач), автомобіль (браузер) і водій таксі [25]. Детальна архітектура технології показана на рис. 1.6.

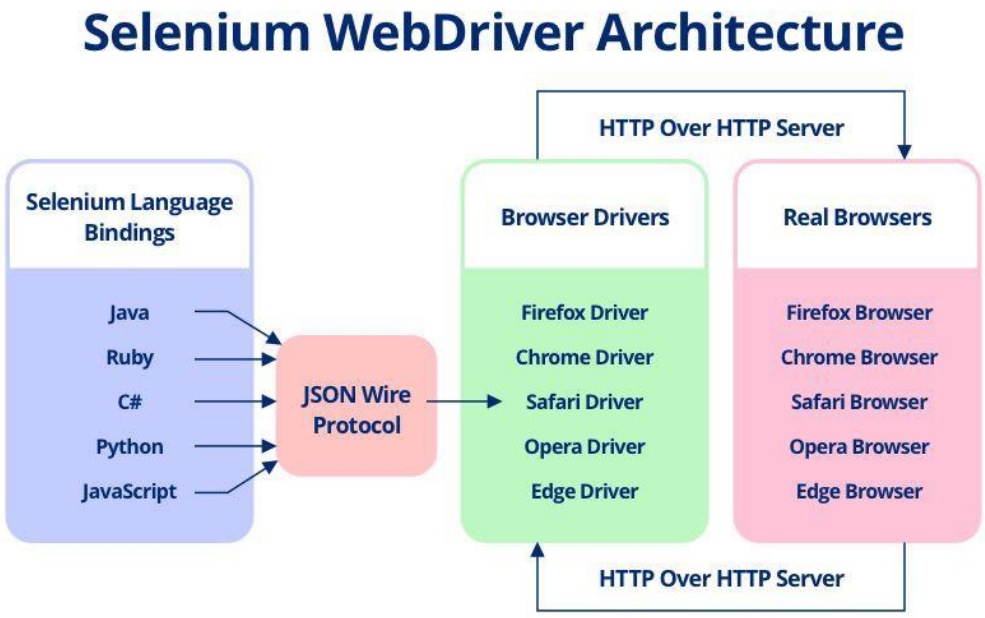


Рисунок 1.6 – Архітектура Selenium WebDriver

TestingWhiz [26] дозволяє автоматизувати тестування веб-продуктів, мобільних додатків, програмних інтерфейсів додатків (API), баз даних і ПО. Розглядаються як регресійні, так і кросбраузерні методи тестування [27]. Інтерфейс програми показано на рис. 1.7.

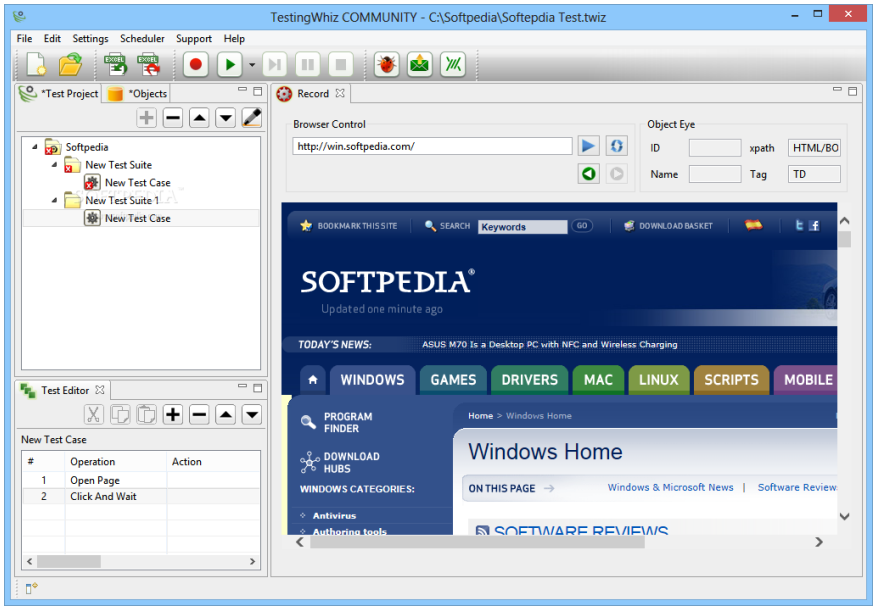


Рисунок 1.7 – Користувацький інтерфейс додатку TestingWhiz

Ліцензований інструмент Hewlett Packard HPE UFT [28]. Уніфіковане функціональне тестування Hewlett Packard Enterprise (HPE UFT) — це програмне забезпечення, яке використовується для автоматизованого функціонального тестування та регресійного тестування програмних додатків. Він раніше називався HP QuickTest Professional (QTP), але тепер називається Hewlett Packard Enterprise (HPE) [29]. використовується для функціонального тестування програмних продуктів. Розпізнає смарт-об’єкти та керує текстом скрипта безпосередньо під час дії користувача завдяки вбудованому механізму обробки багів. Інтерфейс програми показано на рис. 1.8.

Розробники та тестери програмного забезпечення часто використовують HPE UFT, щоб автоматизувати тестування та гарантувати високу якість продуктів.

Підтримка мов програмування, таких як VBScript, є однією з основних характеристик HPE UFT. Це дозволяє створювати складні тестові сценарії та взаємодіяти з елементами користувацького інтерфейсу програми. Крім того, можливості відтворення дій користувача та реєстрації дозволяють платформі спростити створення та підтримку автоматизованих тестів.

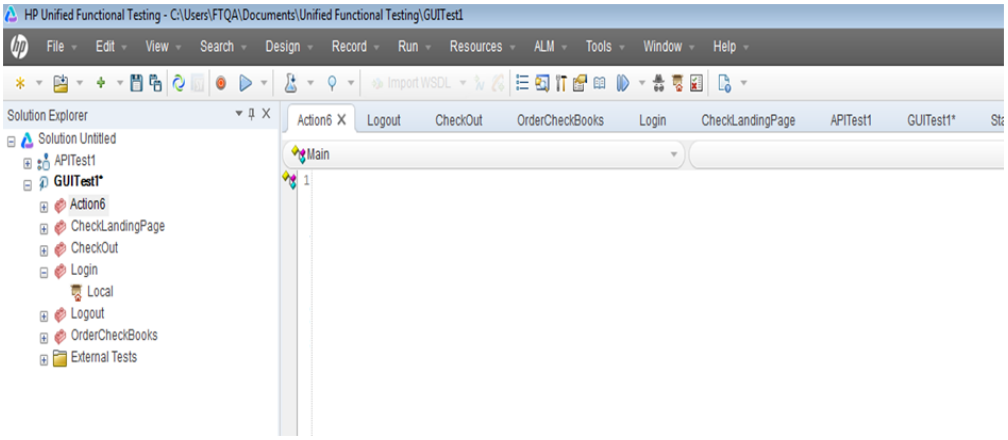


Рисунок 1.8 – Користувацький інтерфейс додатку HPE UFT

JMeter — це програмне забезпечення, розроблене Apache Software Foundation для проведення навантажувального тестування. Спочатку JMeter розроблявся як інструмент для тестування веб-додатків, але зараз він здатний проводити тести навантаження для з'єднань SOAP, JDBC, POP3, FTP, LDAP, JMS, IMAP, TCP та HTTP [30]. Чудова можливість створення великої кількості запитів за допомогою декількох комп'ютерів, використовуючи один комп'ютер для виконання цього процесу. організувати логування результатів тесту та представити їх у різних формах, таких як діаграми, таблиці тощо. Інтерфейс програми наведено на рис. 1.9.

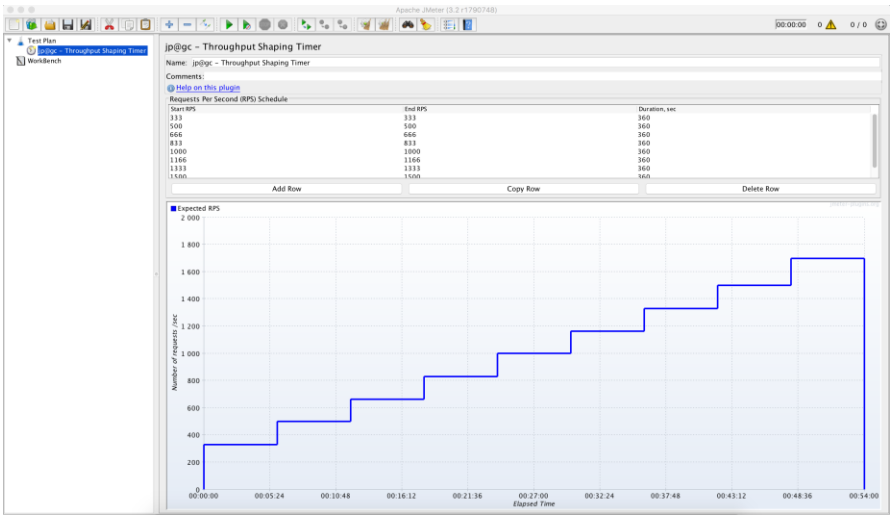


Рисунок 1.9 – Користувацький інтерфейс JMeter

Postman — це інструмент для розробників програмного забезпечення та тестувальників, який допомагає спростити та автоматизувати тестування API.

Postman дозволяє створювати, відправляти та перевіряти HTTP-запити до веб-серверів та інших сервісів із API. Крім того, інструмент дозволяє створювати автоматизовані тестові набори для перевірки API [31].

Postman призначений для відправки запитів на сервер і розробки та тестування програмних інтерфейсів додатків (API) і веб-сайтів. Завдяки власному графічному інтерфейсу можна з легкістю конфігурувати усі потрібні дані для проведення тестів.

Після відкриття колекції будь-який розробник або тестувальник зможе з легкістю дізнатися, як працює сервіс. На додаток до всього іншого, Postman дозволяє розробляти API-дизайн і розробляти Mock-сервери на його основі [32]. На рис. 1.10 наведено інтерфейс програми.

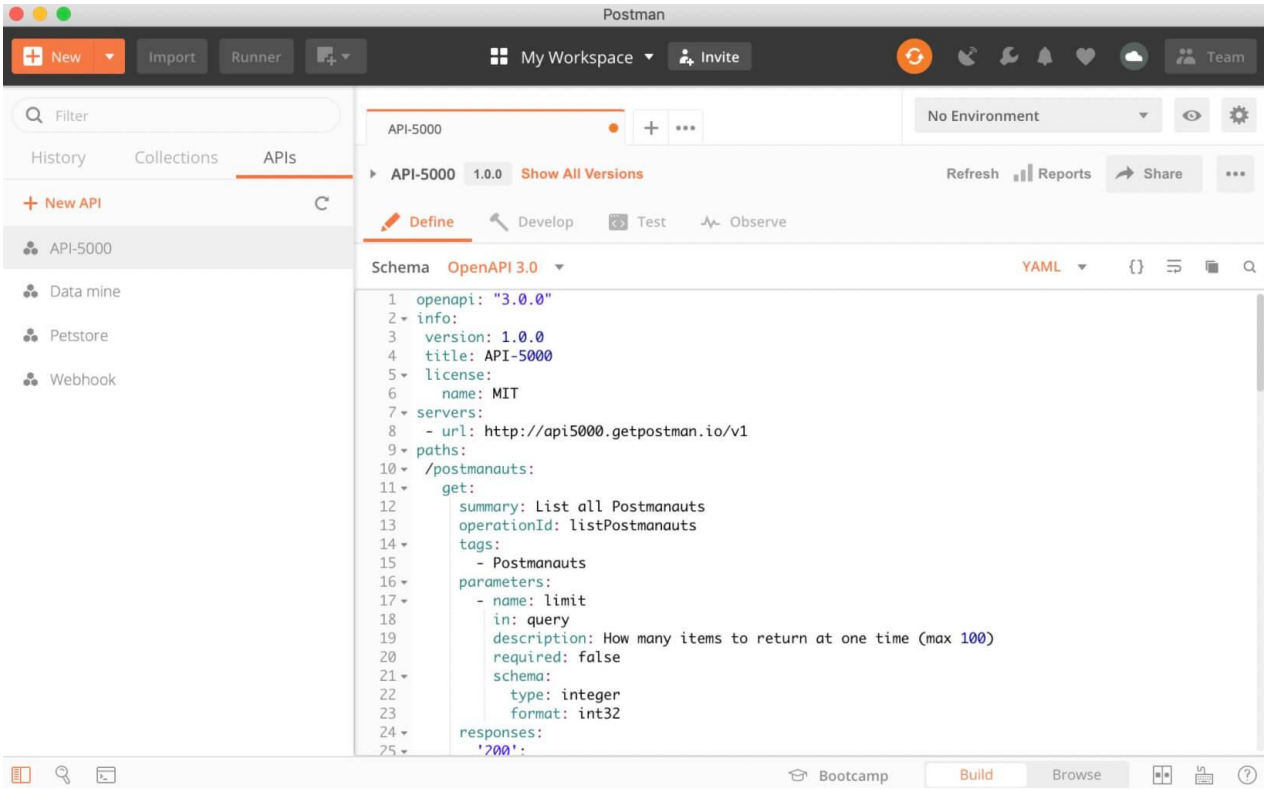


Рисунок 1.10 – Користувацький інтерфейс Postman

У результаті проведеного аналізу та врахування особливостей існуючих підходів для розробки тестових фреймворків, можна сформулювати кілька ключових висновків та напрямків розвитку [33]:

1) Вибір технологій: важливо обрати технології, які найкраще відповідають потребам вашого проекту. Наприклад, якщо ви працюєте з веб-додатками, фреймворки, які підтримують веб-тестування, такі як Selenium або Playwright, можуть бути відмінними виборами.

2) Розширюваність та гнучкість: розробка фреймворку повинна передбачати можливість легкого розширення та адаптації до змін у проекті. Гнучкість у виборі мов програмування та інтеграція з іншими інструментами є ключовими аспектами.

3) Підтримка різних платформ і технологій: фреймворк повинен бути спроектований так, щоб підтримувати різні платформи та технології. Це забезпечить універсальність в застосуванні та покриття різноманітних проектів.

4) Легкість використання: фреймворк повинен бути легким у використанні, щоб розробники та тестувальники могли швидко оволодіти його функціональністю. Наявність документації, навчальних матеріалів та спрощених API може значно полегшити процес навчання.

5) Підтримка паралельного виконання: підтримка паралельного виконання тестів дозволяє прискорити процес тестування та зменшити час виконання, особливо на великих проектах. Це стає важливим у вимогливих до часу проектах та в умовах неперервної інтеграції.

6) Інтеграція з іншими інструментами: фреймворк повинен легко інтегруватися з іншими інструментами, такими як системи керування версіями, CI/CD системи, засоби моніторингу, тощо. Це забезпечить безперервність процесу розробки та тестування.

7) Врахування тестових даних: забезпечення зручного та ефективного управління тестовими даними. Це важливо для забезпечення надійності тестів та можливості легкої модифікації та розширення тестових сценаріїв.

Загальний успіх фреймворку для автоматизації тестування залежить від його спроектування, здатності адаптуватися до змін у проекті та зручності його використання розробниками та тестувальниками [34]. Правильно побудований та налаштований фреймворк стає потужним інструментом для автоматизації тестування в різних проектах.

Після аналізу популярних інструментів автоматизації тестування було визначено їхні переваги й недоліки. Результати занесено до табл. 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних інструментів

Критерій	Selenium	HPE UFT	Testing Whiz	Postman	JMeter
Вибір технологій	1	0	1	1	1
Інтеграція з іншими інструментами	1	1	1	1	0
Підтримка паралельного виконання	1	1	0	0	1
Легкість використання	1	0	1	0	1
Підтримка різних платформ і технологій	1	1	1	1	0
Інтеграція з іншими інструментами	1	1	1	0	0
Врахування тестових даних	1	1	1	1	0
ВСЬОГО	7	5	6	4	3

У результаті проведеного аналізу досліджено особливості розробки фреймворку для автоматизації тестування. Проаналізовано ряд найвідоміших інструментів, котрі призначені для автоматизації тестування. Вирішено розробляти власний фреймворк, базуючись на технології Selenium, оскільки вона

відповідає усім поставленим до неї вимогам та є найкращою серед конкурентів. Також було встановлено ряд завдань, що необхідно досягти при розробці програмного продукту та план їх виконання.

1.3 Аналіз існуючих методологій розробки

Тестово-орієнтована розробка (Test-Driven Development – TDD) та розробка через поведінку (Behavior-Driven Development – BDD) – це дві методології розробки програмного забезпечення, які спрямовані на покращення якості продукту та комунікації в команді [35].

Розробка, керована тестами, вимагає, щоб розробники створювали автоматизовані модульні тести, щоб визначити вимоги до коду перед написанням коду. Тест перевіряє можливість виконання умов. Вони отримують повідомлення про успішне завершення тесту. Проходження тесту свідчить про те, що поведінка, передбачувана програмістом, дійсна. Розробники часто використовують програмні каркаси для тестування, щоб створити та автоматизувати тестування. Насправді модульні тести охоплюють важливі та непотрібні частини коду [36]. Це може бути код, який часто змінюється, код, який залежить від роботи великої кількості іншого коду або код, який має багато залежностей.

Середовище розробки має якомога швидше реагувати на будь-які зміни коду. Архітектура програми повинна складатися з безлічі сильно пов’язаних елементів, які слабо залежать одне від одного, щоб зробити тестування коду набагато швидшим та простішим.

TDD впливає на дизайн програми, крім перевірки коректності. Тестування дозволяє розробникам швидше визначити функціональність, необхідну користувачеві. Таким чином, до остаточної реалізації рішення деталі інтерфейсу з’являються. Зрозуміло, що стандарти якості коду застосовуються до тестів так само, як і до основного коду [37].

Керована поведінка розробка, яка використовує прості предметно-орієнтовані мови програмування, є продовженням керованих тестів розробки. Ці мови змінюють запити звичайною природною мовою у виконувані тести. Наслідком цього є більш тісний зв’язок між критеріями прийнятності для конкретної функції та тестами, які використовуються для перевірки цієї функції. Це є нормальним наслідком загального тестування TDD [38].

Обидві методології можуть використовуватися для автоматизації тестування. Обирається та впроваджується в залежності від потреб проекту, структури команди та вподобань розробників та тестувальників. Основна різниця полягає в підході до написання та структури тестів.

Нижче розглянемо обидві методології більш детально.

– Тестово-орієнтована розробка (TDD):

Принципи:

1) Перший крок – написання тесту, який описує нову функціональність або виправлення.

2) Другий крок – написання коду, який дозволяє тесту пройти.

3) Третій крок – рефакторинг коду, якщо це необхідно.

Цілі та Вигоди:

+ Покращення реалізації функціональності шляхом ітеративного процесу.

+ Забезпечення високої тестової покриття коду.

+ Зменшення кількості помилок та вартості їх виправлення на пізніших етапах розробки.

Інструменти:

1) JUnit, TestNG (для Java).

2) PyTest, unittest (для Python).

3) NUnit (для .NET).

– Розробка через поведінку (BDD):

Принципи:

1) Опис поведінки системи через "історії" або "сценарії", написані на природній мові.

2) Визначення вимог і поведінки системи у співпраці зі зацікавленими сторонами.

3) Перетворення цих сценаріїв у тести.

Цілі та Вигоди:

+ Покращення співпраці між розробниками, тестувальниками та бізнес-аналітиками.

+ Забезпечення спільного розуміння та визначення функціональності.

+ Покращення читабельності та зрозумілості тестових сценаріїв.

Інструменти:

1) Cucumber (для Java, Ruby, інші).

2) Behave (для Python).

3) SpecFlow (для .NET).

– Схожості між TDD та BDD:

1) Ітеративний процес: обидві методології передбачають ітеративний цикл написання тестів, реалізації коду та рефакторингу.

2) Покращення якості коду: обидві методології спрямовані на покращення якості програмного забезпечення шляхом раннього виявлення помилок.

3) Автоматизовані тести: в обох випадках велика увага приділяється автоматизованому тестуванню.

– Відмінності між TDD та BDD:

1) Мова специфікації: TDD фокусується на письмі тестів у термінах програміста, використовуючи тести відповідно до функціональності. BDD використовує природну мову для опису поведінки та вимог системи [39].

2) Аудиторія: TDD спрямований на розробників та тестувальників. BDD призначений для співпраці між зацікавленими сторонами (бізнес-аналітики, клієнти).

3) Вимоги: TDD створює тести на рівні функціональності. BDD створює тести, які визначають бізнес-поведінку системи.

4) Інструменти: хоча обидві методології можуть використовувати подібні інструменти (наприклад, Cucumber), але їх використання може відрізнятися.

Обидві методології можуть бути використані в комбінації або окремо, залежно від потреб проекту та уподобань команди розробників. Обидві спрямовані на досягнення високої якості продукту та співпрацю всіх учасників розробки [40].

Нижче на основі потреб та задач, які необхідно вирішено наведено порівняльну таблицю двух методологій (табл. 1.2):

Таблиця 1.2 – Порівняння методологій розробки

Критерій	TDD	BDD
Час написання тестів	1	0,5
Чіткість та зрозумілість	0,5	1
Ефективність	0,5	1
Рефакторинг та збереження	1	1
Простота у використанні	1	1
Підтримка	1	0,5
Розширюваність	0,5	1
Підсумковий результат	5,5	6

Згідно табл. 1.2, можна зробити висновок, що методологія розробки BDD підходить найкраще серед усіх розглянутих, так як відповідає усім вимогам і задовольняє всі потреби, котрі можуть виникнути у процесі розробки фреймворку. Завдяки BDD можливо досягти високого рівня взаємодії між командами розробників та тестувальників, а також забезпечити відслідковування виконання функціональних вимог протягом усього циклу розробки.

1.4 Обґрунтування вибору мови програмування та середовища розробки

Вибір мови програмування, технологій і засобів розробки є важливими питаннями перед початком розробки програмного забезпечення. Щоб досягти цього, перед початком написання програми потрібно чітко визначити вимоги до неї та визначити, які функції будуть реалізовані.

ООП-мова програмування, яка підтримує Selenium-технологію, необхідна для реалізації програми. Розглянемо такі мови програмування, як C#, Java, Python та JavaScript, які найчастіше використовуються для створення автоматизованих тестів [41].

C# — це сучасна, об’єктно-орієнтована та строготипізована мова програмування, яка дозволяє розробникам створювати безліч безпечних і надійних програм на основі екосистеми.NET. C# належить до широко відомого сімейства мов C і схожий на C, C++, Java та JavaScript. Мовні конструкції для підтримки об’єктно-орієнтованої концепції роботи доступні в C#. Це означає, що C# найкраще підходить для застосування і створення програмних компонентів.

Мова C# була розроблена, щоб використовувати сильні та слабкі сторони інших мов, наприклад, таких як Java і C++. Андерс Гейлсберг, Скотт Вілтамут і Пітер Гольде написали специфікацію мови C#. У програмуванні Андрес Хейлсберг відомий як лідер команди, яка створила Delphi, а також як творець компілятора Turbo Pascal.

Java — це об’єктно-орієнтована мова програмування, яка сильно типізована. Використання Java у всьому світі полегшується бібліотеками з відкритим кодом. Apache, Google та інші організації створили широку кількість корисних бібліотек, що пришвидшує та полегшує розробку програм будь-якої складності. Перш ніж писати власний код, часто варто пошукати допоміжні бібліотеки в Google. Існує висока ймовірність того, що подібна функція уже розроблена, тестована та доступна для використання.

Гнучка система безпеки, яка повністю контролює виконання програми віртуальною машиною, є ще однією важливою характеристикою технології Java. Повноваження програми обмежені будь-якою операцією. Крім того, методи перевірки автентичності, які базуються на шифруванні з відкритим ключем, додають безпеки.

Java була розроблена для розподіленого програмування, оскільки її програми та дані можуть працювати разом на кількох комп'ютерах, що підвищує продуктивність і ефективність роботи.

Python — це мова програмування високого рівня, яка інтерпретує об'єкти та має строгу динамічну типізацію. Вона дуже популярна серед розробників через простий і читабельний синтаксис. Основною метою цієї мови програмування є підвищення продуктивності розробників за допомогою зрозумілості документації та простоти коду. Це є досить скриптова мова, що може вирішувати багато речей. Python найчастіше використовують для роботи з великими даними, розробки сайтів і мобільних ігор. Він також підходить для розробки мобільних і десктопних програм.

Python використовується в багатьох областях, таких як веб-розробка, наукове моделювання, аналіз даних, штучний інтелект і робототехніка. Його широкий спектр можливостей і простота використання роблять його все більш популярним [42].

JavaScript (JS) — це динамічна, об'єктно-орієнтована мова прототипування. Найчастіше застосовується при створені сценаріїв веб-сторінок. Розробка відбувається відповідно до стандарту ECMAScript. Ці сценарії дозволяють користувачеві (пристрої кінцевого користувача) взаємодіяти з веб-сторінкою, асинхронно обмінюватися даними з сервером, керувати браузером, змінювати структуру та вигляд веб-сторінки. Також JavaScript класифікують як скриптову, прототипну (підмножина об'єктно-орієнтованої) мову програмування з динамічною типізацією.

Окрім прототипної, JavaScript частково підтримує інші парадигми програмування, а також деякі відповідні архітектурні особливості. Зокрема, автоматичне керування пам’яттю, функції як об’єкти першого класу, динамічна та слабка типізація та автоматичне керування пам’яттю.

Порівняння мов програмування наведено в табл. 1.3.

Таблиця 1.3 – Порівняння мов програмування

Критерій	C#	Java	JavaScript	Python
Об’єктно-орієнтованість	1	1	0.5	1
Підтримка Selenium WebDriver	0.5	1	1	0.5
Інструменти для легкого аналізу даних	1	1	1	0,5
Обширність та доступність документації	0.5	1	0,5	1
Простота синтаксису	1	1	0.5	0.5
Підсумковий результат	4	5	3,5	3,5

Базуючись результатів у табл. 1.3, можна зробити висновок, що мова програмування Java найкраще підходить серед усіх розглянутих мов, так як вона задовольняє усі потреби, котрі можуть виникнути у процесі розробки фреймворку. Вона має зрозумілий синтаксис та є простою у використанні, а також, підтримує роботу із технологією Selenium WebDriver.

Вибір інтегрованого середовища розробки є не менш важливим етапом, ніж вибір мови програмування, на якій буде реалізована програма.

Комплексне програмне рішення, яке використовується для розробки програмного забезпечення, називається інтегрованим середовищем розробки (IDE). IDE зазвичай включає інструменти для автоматизації складання та відлагодження програм, а також редактор початкового коду. Розглянемо найвідоміші інтегровані середовища розробки Java, такі як Eclipse та IntelliJ IDEA, які використовуються для розробки програмних продуктів.

Eclipse — це вільне середовище розробки програмного забезпечення, яке підтримує модулі та інтегрується. Платформа Eclipse, набір інструментів Java для програмістів, системи контролю версій, конструктори GUI та інші проекти створені та підтримуються Eclipse Foundation. Оновлений в основному на Java, він може використовуватися для створення програм на інших мовах програмування, таких як Ada, C, Python, C++, COBOL, C#, Perl, PHP, Fortran, R, Ruby, Scala, Clojure та Scheme.

Також слід згадати й Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse JDT для Java, Eclipse CDT для C/C++ та Eclipse PDT для PHP. Усі вони є окремими середовищами розробки. Eclipse — це потужний редактор коду, який має визначення синтаксису, автодоповнення коду, інтегрований відладчик та інші корисні функції. Eclipse має велику спільноту користувачів, яка активно працює. Користувачі можуть оптимізувати робочий процес за допомогою різноманітних розширень, бібліотек і плагінів.

Eclipse має вільний код, широку підтримку багатьох мов програмування та платформ, а ще також велику кількість плагінів, доступних для використання. Проте, інтерфейс Eclipse може вважатися менш інтуїтивно зрозумілим у порівняні з іншими IDE, і він може вимагати деякого часу для налаштування та оволодіння користувачем.

Загалом, завдяки своїй гнучкості, розширюваності та широкому спектру функцій Eclipse є популярною та потужною IDE для розробки Java та інших мов програмування.

IntelliJ IDEA — це середовище розробки для бізнесу, яке включає кілька мов програмування (Java, Python, Scala, PHP тощо) розроблене JetBrains. Система поставляється у вигляді безкоштовної, але урізаної по функціональності версії «Community Edition» і комерційної повнофункціональної версії «Ultimate Edition». Активні розробники відкритих проектів мають можливість одержати безкоштовну повну версію продукту. Всі сирцеві тексти Community-версії поширюються у

рамках ліцензії Apache 2.0. Усі бінарні збірки підготовлені для Mac OS X, Linux і Windows [42].

Інтелектуальна система автоматичного завершення коду IntelliJ IDEA є великою перевагою, оскільки вона дозволяє розробникам писати код швидше та зменшити ймовірність помилок. Велика кількість плагінів і розширень є ще однією важливою характеристикою, яка робить його надзвичайно розширюваним і гнучким.

IntelliJ IDEA володіє рядом функцій, які полегшують і прискорюють розробку програмного забезпечення. Його редактор коду підтримує автоматичне завершення коду, рефакторинг, аналіз коду на льоту і вбудований відладчик, що робить процес програмування більш зручним і продуктивним [44].

У спільної версії середовища IntelliJ IDEA доступні плагіни для тестування TestNG і JUnit, засоби складання Ant, Gradle, Maven, системи контролю версій Subversion, Mercurial, CVS та Git, а також мови програмування Scala, Clojure, Java, Groovy і Dart. Також підтримується розробка додатків для платформи Android [33]. До комплекту входять XML-редактор, редактор регулярних виразів, модуль візуального проектування GUI-інтерфейсу Swing UI Designer, система перевірки коректності коду, доповнення для імпорту та експорту проектів через Eclipse й система контролю за виконанням завдань. Засоби для інтеграції з системами відстеження помилок, такі як JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack і Lighthouse, доступні [45].

Великий обсяг ресурсів, використовуваних середовищем, може призвести до проблем, які можуть вплинути на роботу менш потужних систем. Інтерфейс також може здатися складним для деяких користувачів на початковому етапі.

Загалом IntelliJ IDEA відомий своєю високою продуктивністю, багатофункціональністю та приємним інтерфейсом. Розробники Java та інших мов програмування часто використовують його для створення різних типів програмних проектів [46].

Visual Studio — це IDE (інтегроване середовище розробки) від компанії Microsoft, яке пропонує широкий набір інструментів, які дозволяють створювати різноманітні програмні продукти. Visual Studio пропонує широкий спектр важливих характеристик і функцій, включаючи можливості для створення десктопних, веб-, мобільних і хмарних додатків [47].

Зокрема, Visual Studio має потужний редактор коду, який підтримує функції підсвічування синтаксису, IntelliSense для автодоповнення коду, а також зручний навігатор коду. Інтегрований відладчик дозволяє відстежувати та виправляти помилки, встановлювати точки зупинки та проводити детальний аналіз змінних [48].

У Visual Studio можна використовувати широкий спектр мов програмування, таких як C++, Visual Basic, C#, F#, Python та інші. Він є чудовим інструментом для розробників через велику кількість інструментів для тестування, версіонування коду, роботи з базами даних та інших сервісів.

Веб-додатки, мобільні програми та десктопні програми є одними з різноманітних типів застосунків, які можна розробляти з Visual Studio. Можливість створення та відлагодження Windows-служб забезпечує повний досвід розробки Windows. Унікальна система тестування дозволяє швидко перевірити функціональність програми [49].

У Visual Studio багато переваг, таких як широкий доступ до конфігурації, інтеграція з іншими продуктами Microsoft, такими як Azure та Team Foundation Server, а також активна спільнота користувачів. Включені інструменти для роботи з SQL Server, MySQL та іншими, щоб розширити можливості роботи з базами даних. Включені інструменти для статичного аналізу коду допомагають знайти потенційні проблеми та покращити якість коду.

Недоліками можуть бути великий обсяг ресурсів, необхідних для роботи, а також складність для новачків, особливо коли вони починають знайомитися з інтерфейсом і функціями [50].

Усе враховуючи, Visual Studio залишається одним з провідних інструментів для розробки програмного забезпечення, забезпечуючи розробникам різноманітні можливості та інструменти для ефективного та продуктивного творення програм.

Порівняння середовищ розробки наведено в табл. 1.4.

Таблиця 1.4 – Порівняння середовищ розробки

Критерій	Eclipse	IntelliJ IDEA	Visual Studio
Підтримка Java	1	1	0
Підтримка Selenium	0,5	1	0
Обширність та доступність документації	0,5	1	1
Можливості базового функціоналу	1	1	1
Простота у використанні	0,5	1	0,5
Підсумковий результат	3,5	5	2,5

Базуючись на результатах з табл. 1.4, можна зробити висновок, що середовище розробки IntelliJ IDEA найкраще підходить серед усіх розглянутих. Воно покриває всі потреби, які можуть виникнути в процесі розробки фреймворку. Воно є зрозумілим і простим у використанні, а також підтримує роботу із мовою програмування Java та технологією Selenium WebDriver.

1.5 Обґрунтування задач МКР

Обґрунтування задач роботи є досить важливим етапом. Це дозволяє чітко визначити мету та обсяг дослідження, виокремити актуальні питання та внести

свій внесок у вибрану галузь. Головною метою роботи є покращення системи керування навчальним процесом JetIQ за допомогою автоматизації тестування. Тому, серед головних задач слід зазначити наступні:

1) Автоматизація тестового процесу

2) Автоматизація запуску тестів

3) Виконання тестів в кількох браузерах

4) Виконання тестів на віддаленій машині

5) Виконання тестів на локальній машині

6) Зберігання результатів тестування у хмарному сховищі

7) Генерація тестових звітів

Також слід зазначити ряд другорядних задач, таких як:

1) Покращення якості продукту: автоматизоване тестування дозволить виявляти та виправляти помилки та дефекти програмного забезпечення швидше та ефективніше, що призведе до покращення якості системи JetIQ.

2) Зниження витрат часу та ресурсів: автоматизоване тестування дозволить здійснювати тести швидше та безпомилково, що скоротить час, необхідний для ручного тестування та ручної перевірки.

3) Забезпечення надійності: автоматизація тестування дозволить виконувати однакові тести на різних етапах розробки та після змін в коді, що забезпечить надійність системи JetIQ.

4) Забезпечення сумісності: автоматизоване тестування дозволить перевірити сумісність системи JetIQ з різними операційними системами, браузерами та пристроями, що зменшить ризик несправностей на різних платформах.

5) Зменшення людського фактору: ручне тестування піддається помилкам і може бути нестабільним через різні інтерпретації інструкцій. Автоматизовані тести виконуються однаково і не піддаються втомі чи відволіканню.

6) Покращення покриття тестування: завдяки автоматизації можна виконувати більше тестів у відносно короткий час і покривати більше можливих сценаріїв взаємодії з системою.

7) Заощадження ресурсів: зменшення витрат на тестування і підтримку системи завдяки автоматизації.

8) Підвищення ефективності розробки: автоматизація тестування вимагає менше ручного втручання, що дозволить розробникам більше часу приділяти розробці нового функціоналу.

9) Відкриття можливості для неперервної інтеграції та постійної доставки: автоматизоване тестування інтегрується з процесом неперервної інтеграції (CI) та постійної доставки (CD), що дозволить швидше та безпечніше впроваджувати зміни в системі JetIQ.

10) Підвищення задоволення користувачів: завдяки автоматизованому тестуванню, якість системи JetIQ буде вищою, що призведе до більшого задоволення користувачів та збереже їхню довіру.

Зазначені обґрунтування вказують на важливість автоматизації тестування для покращення продуктивності та якості системи керування навчальним процесом JetIQ і обґрунтовують необхідність проведення відповідних досліджень та розробки автоматизованих тестів.

1.6 Обґрунтування новизни роботи

Загальна мета інноваційної роботи полягатиме в тому, щоб забезпечити високу якість та стабільність системи керування навчальним процесом JetIQ і забезпечити задоволення користувачів системи.

Інноваційність роботи полягатиме у впровадженні сучасних підходів та технологій для автоматизації тестування системи керування навчальним процесом JetIQ, що включатиме в себе наступне:

1) Використання інструментів автоматизованого тестування: розробка та інтеграція спеціалізованих інструментів і фреймворків для автоматизації тестування системи JetIQ, які дозволяють швидко та ефективно виконувати тестові сценарії.

2) Скриптова автоматизація: створення скриптів і автоматизованих тестів, які можуть виконувати повторювані операції, тестувати різні аспекти системи та відслідковувати помилки.

3) Тестування реальних умов експлуатації: врахування реальних сценаріїв використання системи JetIQ, зокрема, великої кількості користувачів, навчальних матеріалів, інтерактивності і інших факторів.

4) Використання технологій контейнеризації: використання контейнерів, таких як Docker, для забезпечення ізольованого середовища для тестування, що полегшує налаштування і відлагодження тестів.

5) Автоматизація тестування інтерфейсу користувача: розробка автоматизованих тестів, які перевіряють функціональність та взаємодію з системою через інтерфейс користувача, зокрема за допомогою інструментів, як Selenium або Puppeteer.

6) Розширення покриття тестування: розробка та виконання тестів на всі можливі сценарії використання системи, включаючи тестування різних платформ та пристроїв (крос-платформене тестування).

7) Автоматизована генерація тестових даних: використання інструментів для автоматичної генерації тестових даних та сценаріїв, що допоможе покрити більше можливих варіантів введення даних.

8) Підтримка інтеграції з іншими системами: забезпечення можливості автоматизованого взаємодії системи JetIQ з іншими програмами та системами, які використовуються в навчальному процесі.

9) Звітність і аналітика: розробка інструментів для створення звітів і аналізу результатів тестування, що допомагають виявляти проблеми і покращувати якість системи.

10) Інтеграція з системами збору та аналізу даних: підключення автоматизованих тестів до систем моніторингу та аналізу результатів, щоб оперативно виявляти проблеми і вдосконалювати якість системи.

Ці новаторські підходи та технології допоможуть покращити процес тестування системи керування навчальним процесом JetIQ, зменшити витрати часу та ресурсів, а також забезпечити більшу надійність та якість цієї системи. Застосування новаторських підходів та технологій в процесі тестування системи керування навчальним процесом JetIQ може значно покращити ефективність та надійність цієї системи.

1.7 Висновки

Після проведеного аналізу досліджено особливості автоматизації тестування продукту та розробки тестового фреймворк. Було проаналізовано ряд існуючих інструментів, котрі призначені для автоматизації тестування. Проаналізовано найпоширеніші методології розробки.

В результаті ретельного аналізу було вирішено розробляти власний фреймворк на основі Selenium WebDriver технології, базуючись на BDD методологію розробки. Також було встановлено ряд основних завдань, котрі потрібно досягти під час розробки програмного продукту.

2. АНАЛІЗ СИСТЕМИ КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ JETIQ ТА ЗАДАЧ ЇЇ ТЕСТУВАННЯ

2.1 Аналіз життєвого циклу та моделі розробки системи

Аналіз бізнес-процесів в системі керування навчанням є критично важливим етапом для забезпечення ефективності та вдосконалення навчального середовища. Цей процес включає в себе ретельне вивчення та оцінку всіх етапів навчального циклу, починаючи від планування і закінчуючи оцінкою результатів.

Один із аспектів аналізу полягає в ідентифікації поточних процесів, які включають у себе планування курсів, реєстрацію студентів, ведення розкладу занять, надання навчальних матеріалів, проведення оцінювань та відстеження прогресу студентів. Аналізується, наскільки ці процеси відповідають вимогам програми навчання та потребам студентів.

Далі проводиться оцінка ефективності використання технологій у навчальному процесі, таких як платформи електронного навчання, системи відстеження прогресу студентів та інші інформаційні системи. Аналізується їхній вплив на спрощення доступу до навчального контенту та поліпшення комунікації між викладачами та студентами.

Наприкінці аналізу визначаються можливість вдосконалення та оптимізації існуючих процесів, а також впровадження нових ініціатив для підвищення якості навчання та задоволення потреб учасників освітнього процесу.

Результати аналізу бізнес-процесів слугують основою для стратегічного планування та розробки подальших поліпшень у системі керування навчанням.

Життєвий цикл програмного забезпечення (ПЗ) - це концептуальна модель, що описує різні етапи від початкової розробки ПЗ до його видалення чи припинення використання.

Зазвичай, цей цикл включає наступні етапи:

1) Планування:

– На початковому етапі визначаються мета та обсяг проекту.

– Встановлюються терміни та ресурси для реалізації проекту.

2) Аналіз вимог:

– Збирання та деталізація вимог до програми від користувачів.

– Визначення функціональних та нефункціональних вимог.

3) Проектування:

– Створення архітектури програмного забезпечення.

– Розробка деталей інтерфейсу та структури системи.

4) Реалізація (кодування):

– Написання програмного коду відповідно до визначених вимог та проекту.

– Використання певних методологій розробки, таких як Scrum, Waterfall чи Agile.

5) Тестування:

– Проведення тестів для перевірки відповідності програмного забезпечення вимогам та виявлення помилок.

– Коригування та виправлення помилок.

6) Впровадження та впровадження в експлуатацію:

– Запуск програмного забезпечення в реальне середовище.

– Надання користувачам доступу та підтримка в період впровадження.

7) Підтримка та апгрейд:

– Надання технічної підтримки та виправлення помилок після впровадження.

– Здійснення апгрейдів та додаткового розвитку програмного забезпечення.

Життєвий цикл ПЗ може варіюватися залежно від методології розробки (наприклад, водоспадний, ітеративний, чи Agile), а також від конкретних вимог та особливостей проекту.

Існує кілька моделей розробки програмного забезпечення (ПЗ), які визначають етапи та процеси у розробці програм. Серед них: водоспадна, гнучка, V-подібна, інкремента, спіральна. Кожна з моделей має свої переваги та недоліки. Далі розглянемо дві найпоширені моделі – водоспадну та гнучку.

Модель водоспаду та Scrum – це дві різні методології розробки програмного забезпечення, кожна з яких має свої унікальні особливості та підходи.

Модель водоспаду є лінійною та послідовною методологією, де кожен етап розробки розпочинається лише після завершення попереднього. Вона передбачає жорстку структуру, де вимоги повністю визначаються на початку проекту, а будь-які зміни вимог у середині розробки можуть бути важкими та витратними. Водоспад добре підходить для проектів зі стабільними та чітко визначеними вимогами.

Scrum, навпаки, є гнучкою та ітеративною методологією, спрямованою на швидке видачу робочого програмного продукту через короткі цикли розробки, називані "спрінтами". Scrum покладає акцент на активну комунікацію в команді, невеликі інкрементальні видачі та здатність адаптуватися до змін у вимогах. Це особливо корисно для проектів, де вимоги можуть змінюватися, а комунікація та співпраця важливі для успіху.

Обидві моделі мають як свої переваги, так і недоліки. Тому вибір між ними залежить від характеристик проекту та конкретних вимог. Модель водоспаду може бути ефективною для проектів зі стабільними вимогами, тоді як Scrum може бути більш підходящим для динамічних та змінних сценаріїв розробки.

Оскільки система JetIQ розроблена по гнучкій моделі розробки, далі розглянемо її більш детально.

Scrum - це гнучка та ітеративна методологія розробки програмного забезпечення, яка надає підхід до керування та виконання проектів з високою ступенем несхожості та змінності вимог.

Scrum розробляється через короткі періоди часу, називані "спрінтами", які можуть тривати від 1 до 4 тижнів. Кожен спрінт завершується видачею інкрементальної версії програмного продукту. Це дозволяє швидше отримувати зворотний зв'язок та швидше реагувати на зміни.

Ключовими ролями в Scrum є Scrum Master, Product Owner та Development Team. Scrum Master відповідає за встановлення та забезпечення дотримання правил Scrum, Product Owner визначає вимоги та пріоритети, а Development Team відповідає за розробку та постачання продукту.

Scrum акцентує на активній та відкритій комунікації між всіма учасниками процесу. Це сприяє розумінню вимог, уникненню непорозумінь та швидкій адаптації до змін.

Кожен спрінт починається спрінт-плануванням, де команда обирає завдання, які вона зобов'язана виконати. Після завершення спрінта проводиться ретроспектива, на якій команда аналізує свою роботу та визначає можливі шляхи вдосконалення.

Scrum сприяє адаптивності та високій продуктивності, забезпечуючи прозорість у розробці та сприяючи швидкому реагуванню на зміни вимог чи обставин.

Нижче на рис. 2.1 наведно життєвий цикл системи JetIQ.

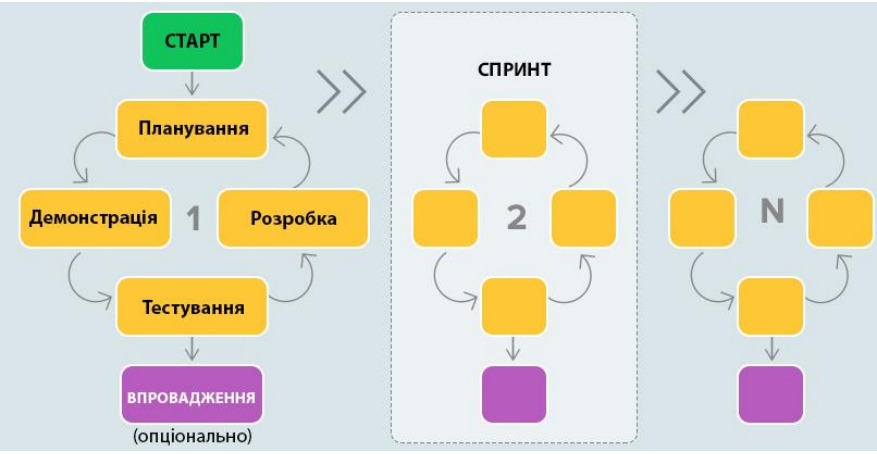


Рисунок 2.1 – Життєвий цикл системи JetIQ

Кожен спринт має декілька етапів, таких як: планування, демонстрація, розробка, тестування та впровадження. Автоматизація етапу тестування є ключовою частиною життєвого циклу програмного забезпечення (ПЗ), яка спрямована на поліпшення ефективності, якості та швидкості розробки.

Такий фреймворк надає можливість автоматизовано виконувати тестові сценарії, спрощуючи процес тестування та зменшуючи час, необхідний для виявлення та виправлення помилок. Розробка такого фреймворку включає в себе кілька ключових аспектів:

Забезпечення зручності вибору технологій та інструментів для розробки фреймворку, враховуючи вимоги проекту та особливості тестування. Створення структури фреймворку, включаючи папки для тестів, конфігураційні файли та бібліотеки.

Розробка механізмів для обробки винятків та створення деталізованих звітів про результати тестів. Інтеграція фреймворку з системою неперервної інтеграції та постачання для автоматичного виконання тестів під час релізів та змін в коді.

Супровід та розширюваність фреймворку для забезпечення легкої адаптації до змін та оновлень. Здійснення тестування автоматизації для перевірки ефективності фреймворку та виявлення можливих недоліків.

Навчання команди розробників та тестувальників та створення документації, яка пояснює структуру та використання фреймворку. Визначення мети тестування та адаптація фреймворку відповідно до змін у вимогах проекту та виявлених помилках.

Отже, розробка власного фреймворку для автоматизації тестування системи керування навчальним процесом JetIQ – це дуже влучне рішення. Ключовою метою є створення потужного інструменту, який не тільки задовольняє поточні потреби проекту, але й залишається гнучким та легко розширюваним для майбутніх вимог.

2.2 Аналіз вимог до системи, розробка тест-кейсів та матриці трасування

У процесі розробки програмного забезпечення важливим етапом є аналіз вимог. У цьому етапі визначаються, документуються, аналізуються та перевіряються вимоги, які стосуються функціональності та характеристик системи. Цей метод допомагає зрозуміти потреби користувачів і як система повинна працювати, щоб вони були задоволені.

Тестування програмного забезпечення (ПЗ) — це процес перевірки відповідності заявлених до продукту вимог і реальної функціональності шляхом спостереження в штучно створених ситуаціях за його роботою й на обмеженому наборі тестів, обраних спеціально. Отже, перш ніж створювати тестові сценарії, потрібно визначити вимоги, які повинні бути протестовані.

Вимоги включають логіку роботи продукту (функціональні вимоги), дизайн (користувацький інтерфейс) і нефункціональні вимоги. Функціональні вимоги часто описуються за допомогою використання сценаріїв використання. У користувацьких сценаріях користувач може взаємодіяти з ПЗ різними способами. Нефункціональні вимоги описують обмеження, пов’язані з розробкою продукту та його виконанням.

Опис логіки роботи у вимогах дозволяє уточнити очікування та функціональність продукту з точки зору розробників, тестувальників, клієнтів та інших зацікавлених сторін. Вимоги допомагають створити чітке та узгоджене розуміння того, як продукт має працювати, і вони служать основою для подальшого проектування, розробки та тестування системи.

Вимоги повинні відповідати наступним характеристикам: ясність, реальність, правильність, необхідність, простежуваність, коректність, узгодженість, модифікованість.

JetIQ – це система керування навчальним процесом у ВНТУ. Головною її функцією є взаємодія між студентами та викладачами. Далі складемо табл. 2.1,

яка містить базові функціональні можливості системи, які підлягають тестуванню та автоматизації.

Таблиця 2.1 – Перелік вимог до системи

ID	Назва
B1	Реалізувати та протестувати вхід студента у власний кабінет
B2	Реалізувати та протестувати “Матеріали”
B2.1	Реалізувати та протестувати меню “Матеріали” - “Матеріали моїх дисциплін”
B2.2	Реалізувати меню “Матеріали” - “Матеріали моєї спеціальності”
B2.3	Реалізувати та протестувати меню “Матеріали” - “Матеріали всіх дисципліни”
B2.4	Реалізувати та протестувати меню “Матеріали” - “Мої силабуси”
B3	Реалізувати та протестувати меню “Розклад”
B4	Реалізувати та протестувати меню “Мої результати”
B5.1	Реалізувати та протестувати меню “Повідомлення” - “Написати викладачу”
B5.2	Реалізувати меню “Повідомлення” - “Надіслати файли”
B6.1	Реалізувати та протестувати меню “Смартфон” - “Android”
B6.2	Реалізувати та протестувати меню “Смартфон” - “IOS 14+”
B7	Реалізувати та протестувати вихід студента із власного кабінету

Отже, як результат маємо 13 вимог до системи, які підлягають тестуванню. Кожна вимога має свій власний унікальний ID. Далі необхідно створити низку тестових сценаріїв, котрі будуть покривати ці вимоги. Після того, як будуть створені тест-кейси, розробимо спеціальну матрицю трасуванню, яка покаже покриття вимог тестами.

Документ, який містить послідовність вказівок для тестувальника, щоб провести певне тестування на програмному продукті з метою перевірки певних функцій або властивостей системи, називається тест-кейсом. Процес тестування включає тестування кейсів.

Основним завданням тест-кейсу є визначення найбільш ефективного методу перевірки певних функцій або властивостей програмного забезпечення. Тест-кейс може мати багато атрибутів, серед яких основними є:

1) ID (унікальний номер)

2) Назва;

3) Кроки відтворення;

4) Передумови;

5) Очікуваний результат.

Такий формат тест-кейсу дозволяє ясно визначити, що тестується, як це робиться, та який повинен бути очікуваний результат. Кожен тест-кейс призначений для перевірки конкретної функціональності чи аспекту програмного забезпечення.

Тест-кейси можуть бути:

1) Позитивні (використовуються лише правильні дані й перевіряють, чи правильно виконується функція в додатку);

2) Негативні (використовуються будь-які дані (як коректні, так і некоректні) і ставить за мету перевірку виняткових, переважно неправильних, ситуацій (спрацьовування валідаторів), а також перевіряє, що та чи інша функція не виконується при спрацьовування відповідного валідатора).

Нижче наведено основі стани проходження тест кейсу:

1) Passed (пройдено);

2) No run (не запущено);

3) Blocked (заблоковано);

4) Failed (помилка);

5) Not Completed (не завершено).

Тест-кейси допомагають проводити перевірку продукту без отримання повного ознайомлення з документацією. Крім того, вони дозволяють обчислювати метрики тестового покриття та систематизувати підхід до тестування.

Обов'язкові вимоги до тест кейсів:

1) Відсутність залежності один від одного;

2) Чіткі формулювання та висока ймовірність виявлення помилки;

3) Наявність детальної та не надлишкової інформації;

Тест-комплект — це набір тест-кейсів. Іноді тест-набір і тест-план переплітаються. Тест-план містить інформацію про те, що потрібно зробити для тестування продукту, коли і як це робити, а також що потрібно зробити.

Нижче наведено рис. 2.2 – 2.3, на яких зображено створені тест-кейси для системи керування навчальним процесом JetIQ.

ID	Назва	Передумови	Кроки	Очікуючий результат
T-1	Кабінет Студента - Перевірка меню: Матеріали -> Матеріали моїх дисциплін	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Натиснути на кнопку 'Матеріали'	1. Натиснути на кнопку 'Матеріали моїх дисциплін'	Відобразилась таблиця електронних матеріалів
T-2	Кабінет Студента - Перевірка меню: Матеріали -> Матеріали моєї спеціальності		1. Натиснути на кнопку 'Матеріали моєї спеціальності'	
T-3	Кабінет Студента - Перевірка меню: Матеріали -> Матеріали всіх дисциплін		1. Натиснути на кнопку 'Матеріали всіх дисциплін'	Відобразилась таблиця "Електронні навчальні матеріали"
T-4	Кабінет Студента - Перевірка меню: Матеріали -> Мої силабуси	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента	1. Натиснути на кнопку 'Мої силабуси'	Відобразилась таблця матеріалів з посиланнями на них
T-5	Кабінет Студента - Перевірка меню: Мої Результати -> Індивідуальний план студента		1. Натиснути на кноку 'Індивідуальний план студента'	Відкрилась сторінка із індивідуальним навчальним планом студента

Рисунок 2.2 – Тест-кейси T-1 – T-5

ID	Назва	Передумови	Кроки	Очікуючий результат
Т-6	Кабінет Студента - Перевірка меню: Мої Результати -> Електронний журнал успішності	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Натиснути на кноку 'Мої Результати'	1. Натиснути на кноку 'Електронний журнал успішності'	Відкрилась сторінка із електронним журналом успішності
Т-7	Кабінет Студента - Перевірка меню: Мої Результати -> Журнал пропусків та заборгованостей		1. Натиснути на кноку 'Журнал пропусків' 2. Повернутись назад і натиснути на кноку 'Журнал заборгованостей'	1. Відкрилась сторінка із журналом пропусків 2. Відкрилась сторінка із журналом заборгованостей
Т-8	Кабінет Студента - Вихід з власного кабінету	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента	1. Натиснути на кноку 'Вихід'	Студент вийшов з власного кабінету
Т-9	Кабінет Студента - Перевірка інтерфейсів відправки файлів та повідомлень викладачу	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Натиснути на кнопку 'Повідомлення'	1. Натиснути на кнопку 'Відправити повідомлення' 2. Повернутись на вкладку назад і натиснути на кнопку 'Надіслати файл'	1. Відобразився інтферейс 'Повідомлення викладачу' 2. Відобразився інтферейс 'Надіслати файл
Т-10	Кабінет Студента - Відправка пустих повідомлень та повідомлень без файлів викладачу		1. Натиснути на кнопку 'Відправити повідомлення' 2. Натиснути на кнопку 'Відправити' 3. Повернутись на вкладку назад і натиснути на кнопку 'Надіслати файл' 4. Натиснути на кнопку 'Відправити'	1. Відобразився інтферейс 'Повідомлення викладачу' 2. Відобразилось повідомлення про помилку відправки пустого повідомлення 3. Відобразився інтферейс 'Надіслати файл викладачу' 4. Відобразився попап із помилкою відправки файла
Т-11	Кабінет Студента - Перевірка меню: Смартфон -> Android та IOS	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Навести на кнопку 'Смартфон'	1. Натиснути на кнопку 'Андроїд' 2. Повернутись назад та натиснути на кнопку 'IOS'	1. Відкрилась сторінка додатка у Play Market 2. Відкрилась сторінка додатка у Apple Store
Т-12	Кабінет Студента - Перевірка меню: розклад	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента	1. Натиснути на кнопку 'Розклад'	Відобразилась таблиця 'Розклад'

Рисунок 2.3 – Тест-кейси Т-6 – Т-12

Отже, створено 12 тест-кейсів, кожен з яких має свій унікальний ID та покриває одну з найголовніших функцій системи керування навчальним процесом JetIQ – кабінет студента. Далі розглянемо детальніше, які ще додаткові функції покривають тест-кейси:

1) Т-1, Т-2, Т-3, Т-4 – покривають функціонал у меню “Матеріали”.

2) Т-5, Т-6, Т-7 – покривають функціонал у меню “Мої Результати”.

3) Т-8 – покривають функціонал виходу користувача (студента) із власного кабінету.

4) Т-9, Т-10 – покривають функціонал взаємодії студента із викладачем (відправка файлів, листування тощо).

5) Т-11 – покривають функціонал у меню “Смартфон”.

6) Т-12 – покривають функціонал у меню “Розклад”.

Тепер потрібно впевнетись у тому, що кожен тест-кейс покриває хоча б одну вимогу і що не існує такої вимоги, яка не була б протестована. Для цього потрібно розробити спеціальну матрицю трасування.

Матриця трасування (RTM) – це двовимірна таблиця, яка містить відповідність функціональних вимог продукту (функціональних вимог) і підготовлених тестових сценаріїв (тестових сценаріїв). QA-інженери використовують матрицю трасування вимог, щоб переконатися, що продукти відповідають вимогам тестів.

Матриця трасування є важливим інструментом для управління проектами та розробки програмного забезпечення. Основна мета RTM полягає в тому, щоб встановити та підтримувати чіткі зв’язки між різними етапами процесу розробки та вимогами, визначеними для проекту. Ця матриця має кілька ключових переваг, включаючи системний підхід до відстеження та керування вимогами протягом усього життєвого циклу проекту.

Одна з основних функцій RTM полягає у відстеженні вимог від їхнього початкового визначення до завершення проекту. Це забезпечує переконаність, що всі визначені вимоги враховані та реалізовані, запобігаючи можливим пропускам

чи непорозумінням. Матриця дозволяє визначити повноту реалізованих вимог, що корисно для уникнення недоліків та непорозумінь.

RTM є важливим інструментом управління змінами, коли виникають зміни вимог проекту, допомагаючи відстежувати ці зміни та оцінювати, як вони впливають на весь проект. Ця можливість дозволяє командам дотримуватися вимог, адаптуючись до зростаючих потреб проекту.

Матрица трасування показує, як взаємозв’язані вимоги та тест-кейси. На перетині відповідних рядків і стовпців ставиться відмітка, щоб показати, що тест-кейс задовольняє цю вимогу. Таким чином, таблиця показує два параметри: наявність в системі вимог, які ще не покриті; це означає, що вимоги не мають жодного перетину з тест-кейсами; або є надмірне тестування в системі, що означає, що вимоги мають кілька перетинів з тест-кейсами.

Матриця трасування дозволяє:

1) візуалізувати актуальний стан реалізації;

2) розбивати вимоги на більш атомарні і структурувати їх;

3) відстежувати, чи є такі вимоги, на котрі ще не запланована розробка, тобто пропуск реалізації;

4) відстежувати, чи реалізовано вимогу в даний момент;

5) наочно відображати пріоритезацію вимог;

6) відстежувати, чи покрита вимога тест-кейсом (пропуск тестування).

Існують такі варіанти зв'язків у матриці трасування:

1) 1 до 1 (атомарна вимога, яка покривається лише одним тест-кейсом, тобто даний тест-кейс покриває лише цю одну вимогу);

2) 1 до n (вимога, що покривається декількома різними тест-кейсами, дані тест-кейси покривають лише цю вимогу);

3) n до n (вимога, що покривається декількома різними тест-кейсами, дані тест-кейси покривають це та деякі інші вимоги).

Слід зазначити щодо зв'язку n до n, коли одна вимога в матриці трасування покривається декількома різними тестами, то це як правило говорить про

надмірність тестування. У такому випадку треба проаналізувати, наскільки вимога атомарна. Тестування не буде надмірним, якщо виконання кожного тесту забезпечує повне покриття та тести не дублюються. Крім того, оцінка покриття розраховується окремо для кожного модуля чи фічі.

На основі вимог та тест-кейсів, які наведено вище створимо матрицю трасування та занесемо дані у табл. 2.2.

Таблиця 2.2 – Матриця трасування

ID	B1	B2	B2.1	B2.2	B2.3	B2.4	B3	B4	B5.1	B5.2	B6.1	B6.2	B7
T-1	+	+	+										
T-2	+	+		+									
T-3	+	+			+								
T-4	+	+				+							
T-5	+							+					
T-6	+							+					
T-7	+							+					
T-8	+												+
T-9	+								+	+			
T-10	+								+	+			
T-11	+										+	+7	
T-12	+						+						

Таким чином, базуючись на матриці трасування можна дійти висновку, що кожна вимога покривається як мінімум одним тест-кейсом. Вимоги, які не покриті хоча б одним тест-кейсом, відсутні.

2.3 Розробка архітектури фреймворку

Архітектура тестового фреймворку включає структуру та опис того, як компоненти влаштовані та взаємодіють один з одним. При проектуванні тестового фреймворку немає архітектури, що призводить до численних переписувань коду, що вимагає багато часу. Це призведе до втрати прибутку та незадоволення замовників.

Далі перейдемо до необхідних та важливих частин архітектури будь-якого тестового фреймворку. Умовно її можна поділити на три частини:

1) Тестовий шар, тобто тестові сценарії;

2) Шар імплементації на базі вимог бізнес логіки, тобто реалізація тестових сценаріїв у коді;

3) Ядро, тобто базові функції, тестові ранери тощо.

Приклад базової архітектури наведено на рис. 2.4.

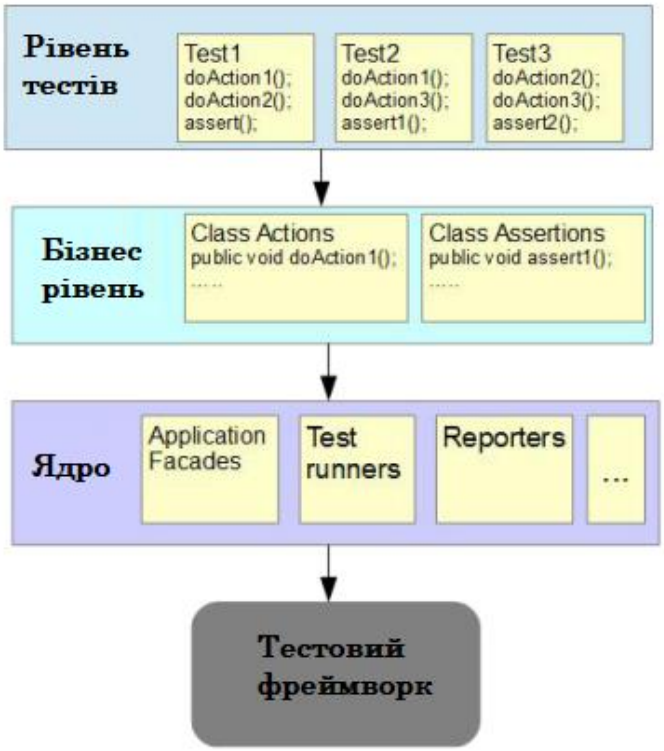


Рисунок 2.4 – Загальний приклад архітектури тестового фреймворку

Вирішено використовувати патерн Page Object, шаблон проєктування, який використовується при написанні автоматизованих тестів. Цей шаблон дозволяє абстрагуватися від відокремлених елементів HTML та інкапсулювати їх в функції доступу до елементів інтерфейсу більш вищого рівня, які користувач може побачити. Це є основою фреймворку Selenium технології. ООП-об'єкт PageObject містить в собі методи, на основі яких генерується DSL для керування застосунком, на основі якої, як правило, пишуть різні варіанти тестування.

Зазвичай сторінка-об'єкт містить лише код для доступу до елементів керування та жодних тестових припущень. Під час створення об'єкта єдиною перевіркою є те, що інтерфейс і елементи керування на ньому завантажені та відображені належним чином.

Головні переваги патерну:

1) Розділення логіки роботи та представлення

2) Зменшення дублювання коду для пошуку елементів керування застосунком

3) При змінах інтерфейсу, що не впливають на логіку, треба лише змінити PageObject, а не логіку тестів

Як було зазначено раніше, у фреймворку будуть представлені тестові сценарії на основі методології BDD. Керована поведінка розробка, яка використовує прості предметно-орієнтовані мови програмування, є продовженням керованих тестів розробки. Ці мови роблять тести на природній мові. Це означає, що існує більш тісний зв'язок між критеріями прийнятності для певної функції та тестами, які перевіряють цю функцію. Це є нормальним продовженням тестування TDD. Методолгія BDD фокусується на наступному:

1) Коли розпочати процеси

2) Що тестувати та що не тестувати

3) Скільки тестувати за один раз

4) Як зрозуміти, чому тести пройшли неуспішно

Зважаючи на ці проблеми, BDD вимагає, аби імена усіх тестів були цілими повними реченнями, котрі починаються з дієслова в умовному способі, та щоб вони відповідали цілям бізнесу. Такий формат імен тестів допомагає зрозуміти, яку саме функцію ви тестуєте та як очікувана поведінка пов’язана з вимогами компанії. Це полегшує спілкування між членами команди розробки та іншими зацікавленими сторонами, такими як тестувальники, менеджери проекту та фахівці з предметної області.

Опис приймальних тестів повинен бути написаний у зручній для користувача мовою. Наприклад, ціле речення в BDD може виглядати так: "Як (роль того, кого бізнес інтереси задовольняються) я хочу, щоб (визначення функціональності так, як вона має працювати), для того щоб (визначення вигоди)". Щоб досягти позитивного результату, критерії приймання мають бути пояснені сценарієм користувача. На рис. 2.5 зображена схема BDD методології.

Рисунок 2.5 – Схема BDD методології розробки ПЗ

На основі цього була розроблена архітектура майбутнього фреймворку, яка поєднує принципи ООП, методологію BDD розкобки ПЗ та патерн-сторінку об’єкта. Схема зображена на рис. 2.6.

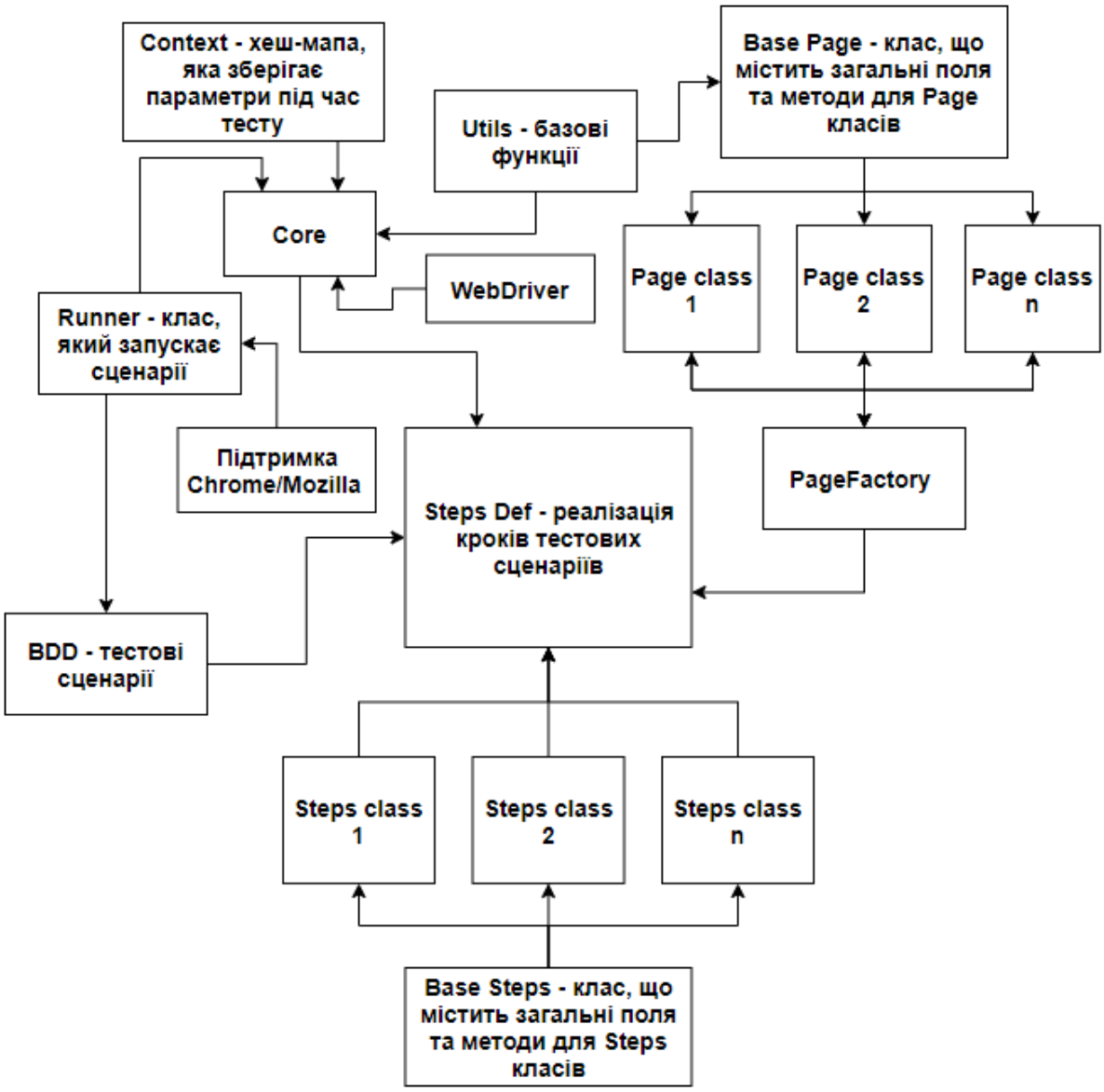


Рисунок 2.6 – Архітектура тестового фреймворку

Таким чином, ми отримуємо загальну архітектурну схему тестового фреймворку, яка відповідає всім вимогам і служить основою для безпосередньої розробки та автоматизації тестування системи керування навчальним процесом JetIQ.

2.4 Розробка алгоритму роботи автоматизованого тестового сценарію

Алгоритм — система правил виконання процесу, котра досягає поставленої мети за скінченний час. Алгоритм складається з набору інструкцій, які вказують, як виконавець має діяти, щоб досягти результату розв’язання задачі за скінченну кількість дій.. Блок-схеми є популярним способом показати алгоритми.

Кожен алгоритм вимагає початкових (вхідних) даних і, як правило, працює, щоб отримати певний результат. Кожен алгоритм виконує свої функції шляхом виконання послідовності кроків. Алгоритмічний процес є процесом виконання цих функцій.

Алгоритм для комп’ютерних програм є списком докладних інструкцій, які реалізують процес обчислення, який починається з початкового стану та відбувається через послідовність логічних станів і завершується кінцевим станом. Деякі алгоритми можуть включати елементи випадковості, щоб перейти з попереднього стану до наступного.

Тестові сценарії та тестові процедури можуть бути включені в тестовий сценарій. Крім того, тестовий сценарій, як і тестова процедура, повинен повертати певне значення, яке точно визначає результати його виконання. Результат виконання будь-якого тестового сценарію є кінцевим результатом, при умові якщо він є основним.

Кожен тестовий сценарій має бути здатний визначати результат його проходження на підставі результатів виконання усіх тестових процедур, котрі він викликає під час виконання.

В той же час, будь-який тестовий сценарій має перевірити виконання умов, необхідних для запуску всіх тестових сценаріїв і процедур, які включені в нього. Умови можуть включати правильність параметрів, переданих тестовому сценарію, наявність необхідних файлів, існування та стан об’єктів, які тестуються, тощо.

Отже, базуючись на наведених вище міркуваннях було розроблено загальний алгоритм роботи тестового сценарію. Блок-схема тестового сценарію зображена на рис. 2.7.

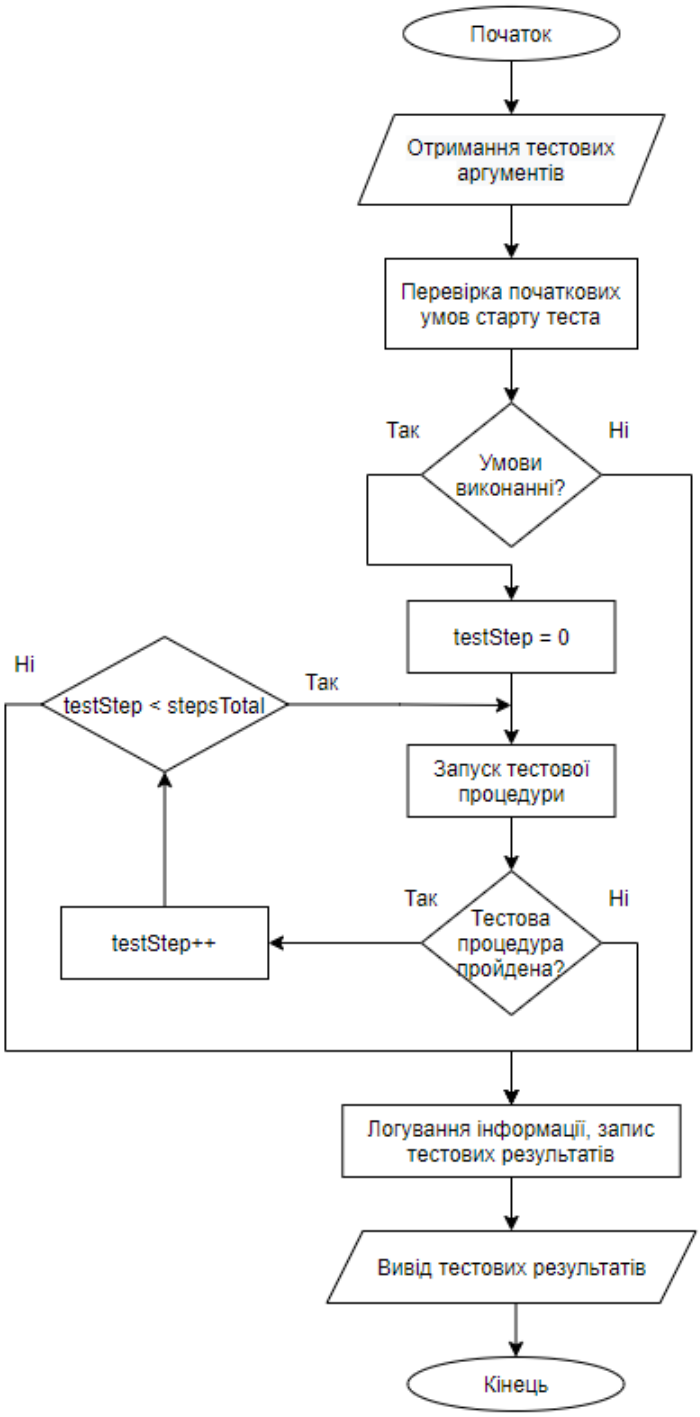


Рисунок 2.7 – Загальна блок-схема алгоритму роботи автоматизованого тестового сценарію

Наступним чином можна описати алгоритм роботи: користувач запускає тестовий сценарій. Тест починається з передачі даних і аргументів. Після цього всі умови, необхідні для початку тестування, перевіряються. Результати виводяться, якщо не виконується хоча б одна з вимог тесту. Відбувається проходження тесту крок за кроком після успішного завершення кожної умови. Якщо не буде виконано хоча б один крок, тест також буде вважатися неуспішним. Логування інформації та вивід тестових результатів користувача відбувається після того, як усі кроки були виконані (або хоча б один не був виконаний). Це останній етап іспиту.

Як результат, розроблено алгоритм роботи тестового сценарію. Порівняно із аналогами, розроблений алгоритм дозволяє писати тестові сценарії українською мовою, зменшити обсяги памяті, що необхідні для тестування та підвищити швидкість і якість тестування. Автоматизовані тестові сценарії будуть розроблятись базуючись на цьому алгоритмі.

2.5 Розробка CI/CD пайплайну

Сучасні методи розробки програмного забезпечення вимагають процесів неперервної інтеграції та неперервної доставки (CI/CD). За допомогою цієї методики можна автоматизувати інтеграцію коду, тестування та розгортання, що гарантує швидку та постійну поставку програмних продуктів.

CI (Continuous Integration) включає в себе регулярне об'єднання коду з репозиторієм, після чого автоматично виконуються процеси збірки та тестування. Це дозволяє вчасно виявляти конфлікти та помилки, забезпечуючи високу якість коду.

CD, також відомі як постійне постачання та постійне розгортання, включають автоматизоване розгортання збірок у тестових або продуктивних середовищах. З безперервним розгортанням потрібно ручне підтвердження перед

запуском, тоді як безперервне розгортання автоматично впроваджує зміни після того, як всі тести пройдені успішно.

Переваги CI/CD включають прискорення розробки, скорочення циклів виправлення помилок, покращення якості коду, зменшення ризиків при розгортанні та забезпечення готовності до впровадження нових функцій. Ці практики стали стандартом в сучасній розробці ПЗ, сприяючи ефективному та стабільному постачанню програмного забезпечення в будь-якому масштабі проекту.

Безперервна розробка, тестування, інтеграція, розгортання та постійний моніторинг програмного забезпечення протягом усього життєвого циклу розробки є прикладами сучасних практик DevOps. Сучасні операції DevOps базуються на концепції CI/CD.

На рис. 2.8 зображено загальну схему CI/CD.

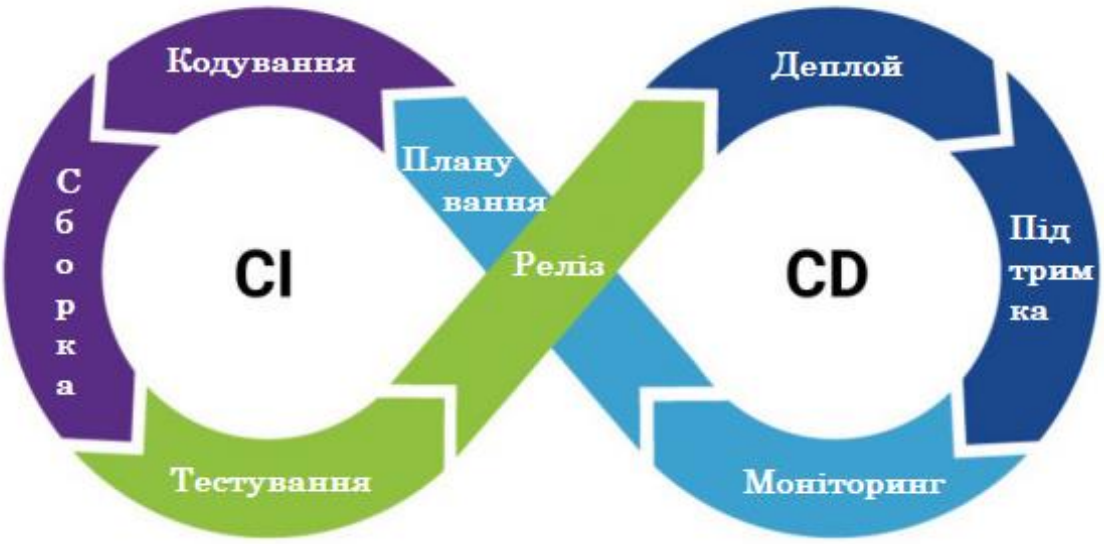


Рисунок 2.8 – Загальна базова схема технології CI/CD

Щоб створити CI/CD, проект потрібно буде підключити до Git системи. Git — це поширена система контролю версій, яка використовується розробниками програмного забезпечення для нагляду та керування змінами в коді. Git,

розроблений Лінусом Торвальдсом, підтримує спільну роботу над проектами та пропонує ефективні можливості керування версіями.

Git зберігає код у вигляді набору версій файлів, також відомих як «знімки», що дозволяє розробникам відстежувати зміни, внесені в різних гілках розробки. Кожна гілка може виконувати унікальний функціонал, і робота в різних гілках дозволяє проводити паралельні експерименти та розробку без впливу на основний код.

На рис. 2.9 зображено схему роботи Git.

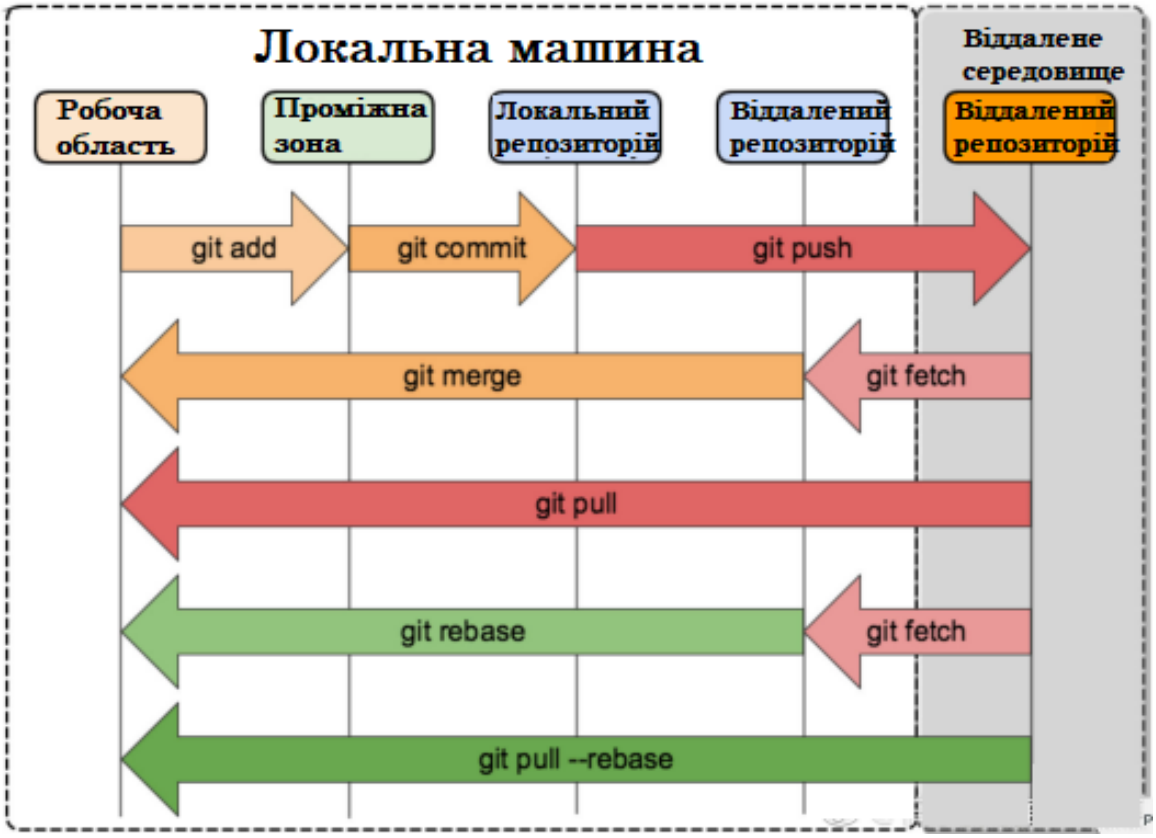


Рисунок 2.9 – Загальна базова схема технології Git

Git є однією з найкращих систем керування версіями, яка пропонує гнучкі інструменти нелінійної розробки, котрі базуються на злитті та відгалуженні гілок. Криптографічні методи використовуються для захисту історії та захисту від змін заднім числом, а цифрові підписи розробників можна прикріпити до тегів і комітів.

Система складається з набору програм, які були розроблені спеціально для використання в скриптах. Це полегшує створення функціональних систем керування версіями на основі Git або GUI. Наприклад, Cogito є фронтендом до репозиторіїв Git. У той же час Git використовується StGit для управління колекцією латок. У системі є кілька інтерфейсів для користування, наприклад, gitk і git-gui розповсюджуються разом із Git.

На основі цього було побудовано CI/CD схему, котра зображена на рис. 2.10, де CI/CD Pipeline – це загальна так звана “труба”, Stage – це стадії пайплайну, а роль джоб виконують круги – саме в них виконуються тести відповідно до спеціальних тегів.

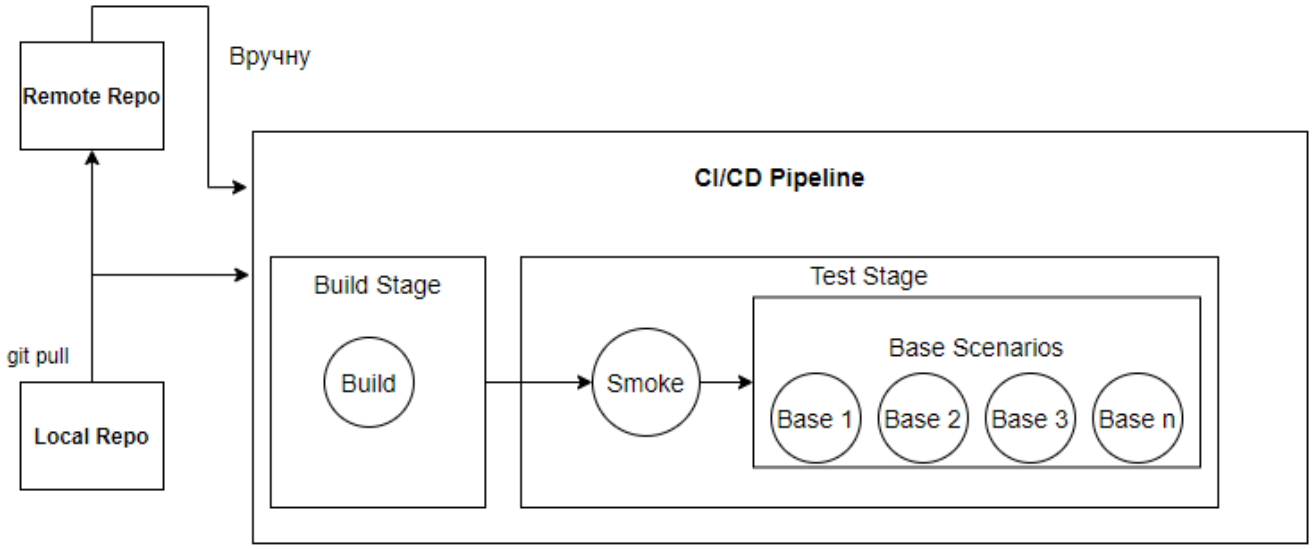


Рисунок 2.10 – CI/CD схема пайплайну для тестового фреймворку

Можна отримати дистанційний доступ до репозиторіїв Git за допомогою git-демона, SSH або HTTP сервера. TCP-сервіс git-daemon є частиною дистрибутива Git і є найпоширенішим і надійним методом доступу разом із SSH. Хоча є деякі обмеження, метод доступу HTTP дуже популярний у контрольованих мережах, оскільки він дозволяє використовувати існуючі конфігурації мережевих фільтрів.

В результаті маємо загальну CI/CD схему для фреймворку, котра відповідає усім поставленим вимогам, містить лише дві стадії: “Build” (збірка проєкту) та “Test” (тестування). Пайплайн запускається при кожному пуші коду до віддаленого репозиторію. Також користувач може й вручну запустити пайплайн на сторінці репозиторію у GitLab, якщо має відповідні права.

2.6 Висновки

У другому розділі наведено аналіз бізнес-процесів та життєвого циклу системи JetIQ. На основі цього аналіз вирішено створити систему тестування, яка дозволить автоматизувати тестування. Було створено ряд вимог для тестування. Крім того, було створено дванадцять автоматизованих тест-кейсів. Створено матрицю трасування на основі вимог і тест-кейсів. Згідно з даними, отриманими з цієї матриці, кожна вимога покривається як мінімум одним тест-кейсом, і відсутні вимоги, які не покриваються жодним тест-кейсом. створено загальну архітектуру тестового фреймворку, яка використовує технології Selenium, патерні Page Object і основні принципи Open Source Programming (ООП). Розроблено схему роботи алгоритму тестового сценарію. Цей алгоритм був використаний для розробки автоматизованих тестових сценаріїв. Крім того, пайплайн CI/CD розроблено та налаштовано для тестування віддалених машин у віртуальному середовищі. В ході розробки CI/CD пайплайну вирішено додати проєкт до GitLab.

3. РОЗРОБКА ФРЕЙМВОРКУ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

3.1 Реалізація тестових сценаріїв за допомогою BDD підходу

У Java для реалізації BDD підходу як правило застосовують Cucumber – середовище автоматичного тестування, котре підтримує BDD (Behavior Driven Development).

Перед інтеграційним або модульним тестуванням етапи тестування та інформація про перевірку заздалегідь оголошуються загальною мовою, щоб всі мали можливість зрозуміти мету кожного етапу. Крім того, розробники тестів інтеграції та модулів можуть використовувати заздалегідь написаний фреймворк для написання коду, щоб досягти мети розробки, заснованої на поведінці.

Cucumber — це відкрите програмне забезпечення, яке використовується для автоматизації тестування програмного забезпечення, написаного на мові програмування Java. Він підтримує специфікації, написані мовою опису Gherkin, яка спрощує спілкування між розробниками, тестерами та іншими членами проекту.

Основна ідея Cucumber полягає в тому, щоб описати функції програми у вигляді природної мови, наприклад англійської, і використовувати цю інформацію для створення автоматизованих тестів. Для сценаріїв тестування можна використовувати «історії користувача», або «історії користувача», які містять конкретні приклади введення та очікувані результати.

Використання ключових слів, таких як Given, When і Then, є однією з ключових характеристик кукурудзи. Ці слова визначають контекст, дію та очікуваний результат відповідно. Це робить тести більш ясними та простими для технічних і нетехнічних учасників проекту. За допомогою коду Java процедури можуть виконуватися автоматично в сценаріях.

Такий метод тестування дозволяє командам співпрацювати та визначати та тестувати функціональність продукту з різних точок зору, що покращує загальну якість програмного забезпечення.

Сценарій тестування міститься у файлах із розширенням .feature. Сценарій можна написати покроково в цьому файлі, а потім ці кроки потрібно реалізувати в коді. Сценарій можна запускати після цього. На рис. 3.1 та 3.2 зображено тестовий сценарій в одному з таких файлів, а також реалізація одного з кроків сценарію у коді відповідно.

```
Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'

  @StudentSchedule
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: розклад
    When Натиснути на кнопку 'Розклад'
    Then Відобразилась таблиця 'Розклад'
```

Рисунок 3.1 – Тестовий сценарій, що написаний за домопогою Cucumber

```
@When("Відкрити домашню сторінку JetIQ")
public void openHomePage() {
    homePage.openHomePage();
    homePage.waitForPageLoadComplete();
    log.info("Home page was opened");
}
```

Рисунок 3.2 – Реалізація одного з кроків сценарію у коді

Таким чином, по такому принципу створено декілька (7) файлів, що містять 12 тест-кейсів, які було представлено у розділі 2.2. Тепер треба створити java класи Steps, котрі будуть реалізовувати кроки з цих файлів у коді.

3.2 Програмна реалізація фреймворку

Тестові сценарії готові, тепер їм потрібно створити реалізацію в коді. Після цього вони можуть вважатися автоматизованими. Код був написаний на Java, відповідно до всіх стандартів ООП і програмування в цілому, з використанням патерну Page Object. Далі наведено частини коду з фреймворку того чи іншого програмного компоненту (рис. 3.4 – 3.9).

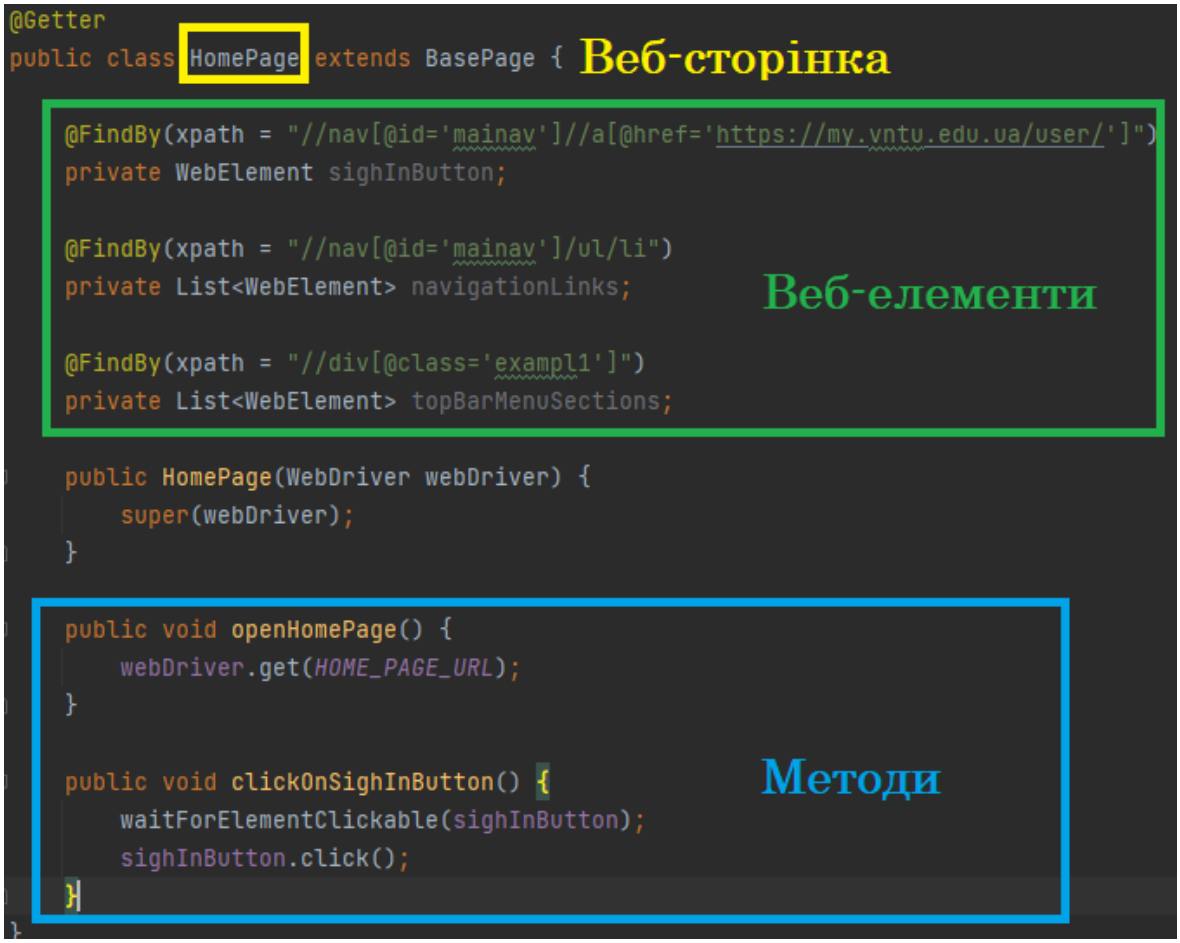


Рисунок 3.4 – Приклад лістингу коду класу веб-сторінки

На рис. 3.4 наведено приклад класу однієї із веб-сторінок. Жовтий означає назву класу, зелений — поля, які представляють веб-елементи (або колекції веб-елементів), які знаходяться на цій веб-сторінці, а синій — способи взаємодії з цими веб-елементами.

```
@Slf4j
public class AssertsSteps extends BaseSteps {

    @Then("Навігаційні посилання відображаються на сторінці")
    public void checkNavLinks() {
        log.info("Start checking navigation links...");
        for (WebElement webElement : homePage.getNavigationLinks())
            assertTrue(homePage.isElementDisplayed(webElement));
        }
        log.info("Navigation links are displayed on page");
    }
```

Рисунок 3.5 – Лістинг коду класу реалізації кроків

Приклад реалізації одного з кроків за допомогою спеціальної анотації @Then, яка належить технології Cucumber, показано на рис. 3.5.

```
@AllArgsConstructor
public class PageFactoryManager {

    private final WebDriver webDriver;

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

    public SignInPage getSignInPage() { return new SignInPage(webDriver); }

    public StudentHomePage getUserHomePage() { return new StudentHomePage(webDriver); }

    public MaterialsPage getMaterialsPage() { return new MaterialsPage(webDriver); }

    public MessagesFilesPage getMessagesFilesPage() { return new MessagesFilesPage(webDriver); }

    public MyResultsPage getMyResultsPage() {
        return new MyResultsPage(webDriver);
    }
}
```

Рисунок 3.6 – Лістинг коду класу, що створює екземпляри класів веб-сторінок

На рис. 3.6 показано приклад реалізації PageFactory патерну, який є удосконаленим видом PageObject патерну. Створення копій класів сторінок здійснюється так званою «фабрикою» цього класу. PageFactory патерн і Factory патерн схожі.

```
Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали -> Матеріали моїх дисциплін
    When Натиснути на кнопку 'Матеріали'
    And Натиснути на кнопку 'Матеріали моїх дисциплін'
    Then Відобразилась таблця електронних матеріалів
    When Обрати диспипліну під номером 1
    Then Відобразилась таблця матеріалів з посиланнями на них

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали -> Матеріали моєї спеціальності
    When Натиснути на кнопку 'Матеріали'
    And Натиснути на кнопку 'Матеріали моєї спеціальності'
    And Обрати будь який семестр
    Then Відобразилась таблця електронних матеріалів
    When Обрати диспипліну під номером 1
    Then Відобразилась таблця матеріалів з посиланнями на них

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали -> Матеріали всіх дисциплін
    When Натиснути на кнопку 'Матеріали'
    And Натиснути на кнопку 'Матеріали всіх дисцпиплін'
    Then 'Електронні навчальні матеріали' table is appeared

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали -> Мої силабуси
    When Натиснути на кнопку 'Мої силабуси'
    Then Відобразилась таблця матеріалів з посиланнями на них
```

Рисунок 3.7 – Реалізація тест-кейсів Т-1 – Т-4

На рис. 3.7 показано приклад реалізації тестових сценаріїв за допомогою Cucumber технології. Кожен сценарій має власний тег.

```java
@SneakyThrows
@After(order = 2)
public void scenarioReporter(Scenario scenario) {
    if (scenario.isFailed()) {
        String fileName = OffsetDateTime.now().toEpochSecond() + ".png";
        log.info("Scenario is failed. Start making a screenshot...");
        File screenshot = ((TakesScreenshot) webDriver).getScreenshotAs(OutputType.FILE);
        FileUtils.copyFile(screenshot, new File( pathname: "target/generated-test-sources/" + fileName));
    }
}
```

Рисунок 3.8 – Лістинг коду методу scenarioReporter

На рис. 3.8 зображено метод scenarioReporter(), який робить скріншот сторінки, у випадку коли сценарій є не пройденим. Мета була розроблена за допомогою функцій Cucumber технології. Скріншот зберігається в папці тимчасових результатів тесту. Цей метод є надзвичайно корисним для пошуку багів і помилок у дебазі.

```java
@Before(order = 2)
public void setUpPageFactory() {
    PageFactoryManager pageFactoryManager = new PageFactoryManager(webDriver);
    scenarioContext.put(HOME_PAGE, pageFactoryManager.getHomePage());
    scenarioContext.put(SIGN_IN_PAGE, pageFactoryManager.getSignInPage());
    scenarioContext.put(USER_HOME_PAGE, pageFactoryManager.getUserHomePage());
    scenarioContext.put(MATERIALS_PAGE, pageFactoryManager.getMaterialsPage());
    scenarioContext.put(MESSAGES_FILES_PAGE, pageFactoryManager.getMessagesFilesPage());
    scenarioContext.put(MY_RESULTS_PAGE, pageFactoryManager.getMyResultsPage());
    log.info("PageFactory was started successfully");
}
```

Рисунок 3.9 – Лістинг коду одного із хуків

Метод setUpPageFactory є одним із хуків. Для того, щоб екземляри класів сторінок було зручно використовувати в майбутньому, цей метод включає їх у контекст.

Хуки використовуються до та після виконання тестового сценарію за допомогою відповідних анотацій: @Before та @After. Змінна «order» вказує на пріоритет виконання хуку. Незалежно від результатів тестування, хуки виконуються завжди (рис. 3.9).

3.3 Налаштування CI/CD пайплайну

Як зазначалося раніше, конфігурація CI/CD потребує включення Git до проєкту. Тому було прийнято рішення створити віддалений репозиторій за допомогою додатку GitLab – системи керування репозиторіями програмного коду для Git. Крім того, він має ряд додаткових можливостей для відстеження помилок, а також власний вікі.

Віддалений репозиторій зображений на рис. 3.10.

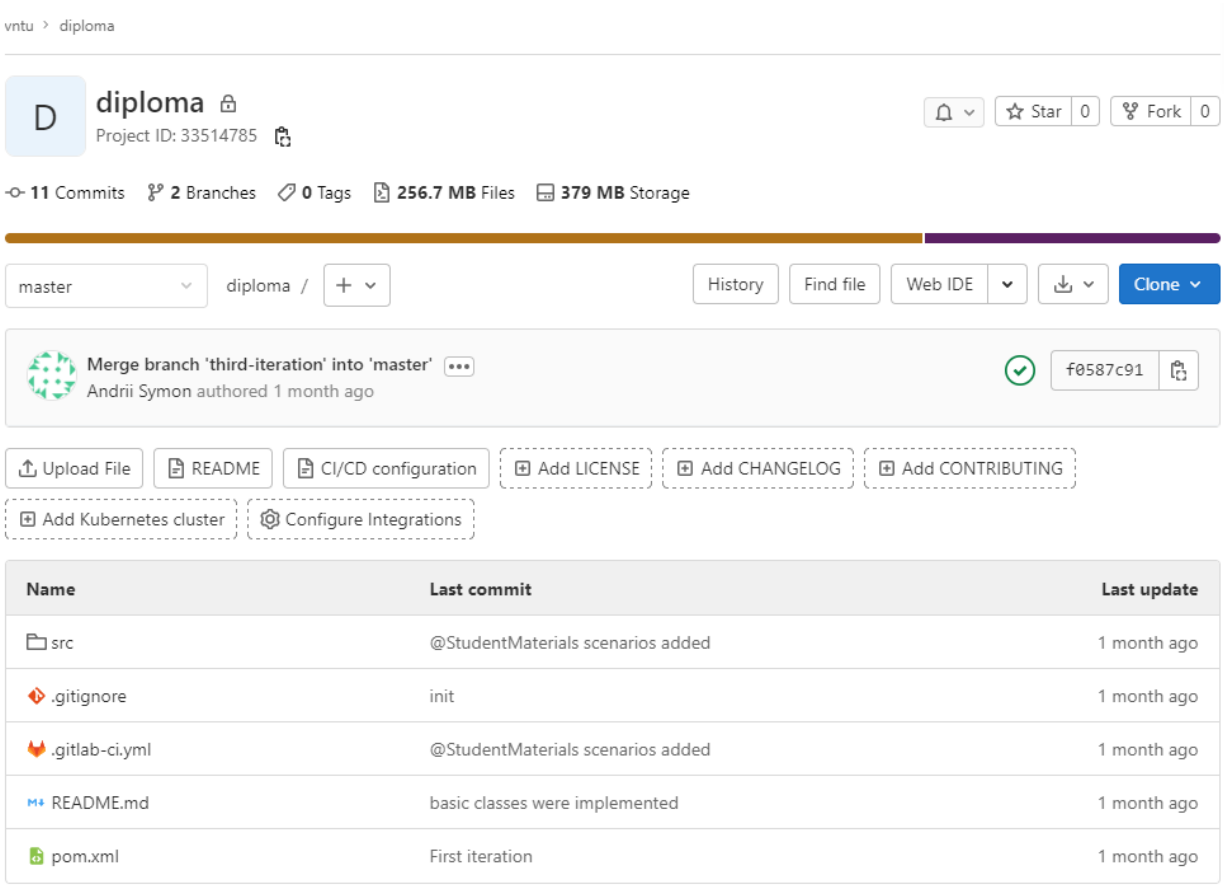


Рисунок 3.10 – Віддалений репозиторій у GitLab

Далі було створено gitlab-ci.yml файл, на базі якого будується CI/CD пайплайн. Тим не менш, перед цим необхідно сконфігурувати віддалену віртуальну машину, оскільки саме на ній проводиться тестування CI/CD. Тому

вирішено застосувати публічний Docker-image “markhobson / maven-chrome: latest” й GitLab Shared Runner на базі ОС Linux.

Далі було створено три стадії пайплайну: Build, Base Test та Smoke Test. Побудова проекту відбувається на першій стадії. Потім відбувається смок тестування, яке включає базові сценарії, без яких будь-яке додаткове тестування неможливе.

Також слід зазначити, що при непроходженні першої чи другої стадії увесь пайплайн зупиняється і помічається як непройдений, тобто “Failed”. Якщо третя стадія непроходить частково або повністю – “Passed with warnings”. Якщо усі три стадії будуть пройдені успішно, то пайплайн помічається як “Passed”.

Конфігурація також включає можливість запускати пайплайн вручну з сайту GitLab. Пайплайн починається автоматично після кожної відправки коміту в віддалений репозиторій.

На рис. 3.11 та 3.12 зображено лістинг-коду, який включає Docker-зображення, стадії, скрипт запуску Cucumber-сценаріїв і джобів, які запускають відповідні сценарії за допомогою певних тегів.

```
image: markhobson/maven-chrome:latest

stages:
  - Build
  - Smoke Tests
  - Base Tests

.run_cucumber_tests: &run_cucumber_tests |
  mvn -f ${CI_PROJECT_DIR}/pom.xml \
  -Dcucumber.filter.tags="${TEST_TAG}" test

Build Project:
  stage: Build
  allow_failure: false
  script:
    - mvn -f ${CI_PROJECT_DIR}/pom.xml -Dmaven.test.skip=true clean package
```

Рисунок 3.11 – Приклад лістингу коду у gitlab-ci.yml файлі, який містить Docker-image, скрипт запуску Cucumber-сценаріїв і стадії

Докер-зображення, яке служить основою контейнеру CI/CD, в якому проводиться тестування, показано на рис. 3.11. Іншими словами, це закрите тестове середовище, заздалегідь налаштоване для віддаленого запуску джоби. Нижче наведено назви трьох стадій.

Скрипт, який запускає тестові сценарії, наведено нижче. Цей скрипт використовує Maven, JUnit і Cucumber одночасно. Нижче наведено зображення джобу Build Project, який відповідає за процес збору проєкту. Поле allow_failure: false вказує на те, що доза не може бути не пройденою в стадії Build.

```
Smoke:
  variables:
    TEST_TAG: "@Smoke"
  stage: Smoke Tests
  allow_failure: false
  script:
    - *run_cucumber_tests

Student Materials:
  variables:
    TEST_TAG: "@StudentMaterials and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests
```

Рисунок 3.12 – Лістинг коду у gitlab-ci.yml файлі, який містить джоби

На рис. 3.12 показано лістинг коду, який створює джоби (Smoke, Student Materials) на CI/CD пайплайні. Оскільки, джоба Smoke містить в собі базові смоук тести, тому вона налаштована так, якщо після її проходження вона матиме статус “Failed”, то пайплайн зупиниться і далі джоби не будуть виконуватись. Це зроблено для економії часу тестувальникі, оскільки, якщо смоук тести мають

статус “Failed”, то і всі інші компоненті та не компоненті тести також матимуть такий статус.

Джоба Student Materials запускає компоненті тести із тегом @StudentMaterials. Статус виконання цієї джоби не впливає на інші джоби та пайплайн вцілому.

GitLab CI/CD пайплайн зображено на рис. 3.13. На рисунку показано пайплайн, стадії та джоби.

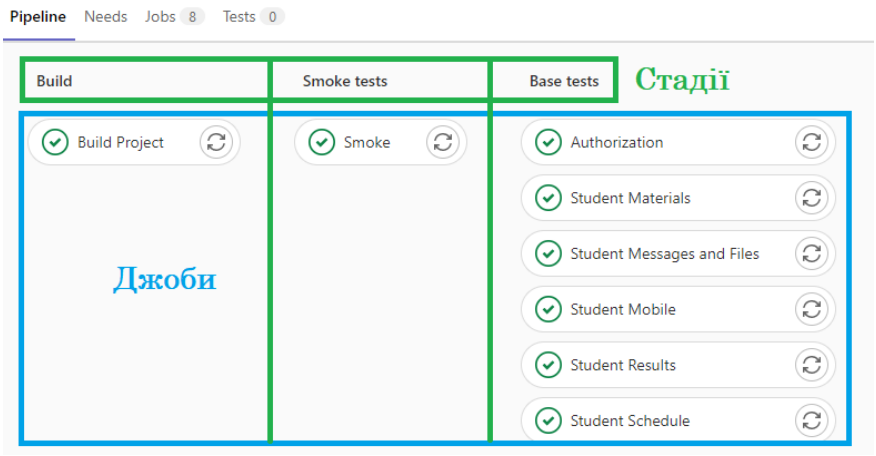


Рисунок 3.13 – GitLab CI/CD пайплайн

Кожен CI/CD пайплайн має унікальний номер, статус, дату виконання, витрачений час на проходження, автора, назву гілки, джоби та стадії. Кожен пайплайн є унікальним та незалежним один від одного. Також при необхідності тестувальник може перезавантажити пайплайн повністю або окремо якусь конкрутну джобу, якщо він має необхідні для цього права.

На рис. 3.14 зображено приклад пройденого пайплайну.

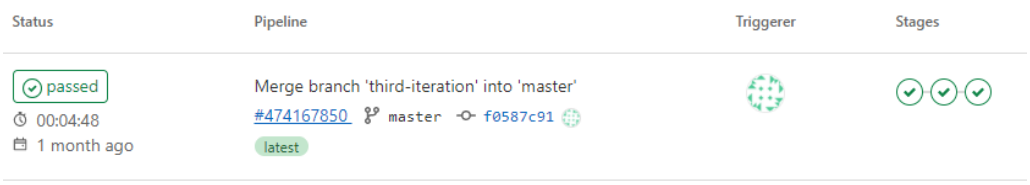


Рисунок 3.14 – Пройдений пайплайн

Як показано на рис. 3.13 та 3.14, пайплайн CI/CD пройшов, що підтверджується зеленим кольором джоба та статусом Passed. Це вказує на те, що всі тести працюють належним чином і що не було виявлено жодних помилок чи багів.

3.4 Висновки

Отже, розробка фреймворку призвела до аналізу та вибору середовища розробки й мови програмування. Було створено автоматизовані тестові сценарії за допомогою технології Cucumber. Було реалізовано програмний додаток на основі технології Selenium та Page Object патерну. Було налаштовано Git та GitLab CI/CD пайплайн для тестування у віртуальному середовищі на віддаленій машині.

4. ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ЗАСТОСУВАННЯ ФРЕЙМВОРКУ

4.1 Інструкція користувача

Створення покрокових інструкцій для користувачів є розумним рішенням, оскільки тестовий фреймворк підтримує тестування як на локальній машині, так і на віддаленій машині за допомогою пайплайну CI/CD. Уся інформація, необхідна для запуску тестів, повинна міститися в інструкції.

Для проведення локального тестування користувач має два способи дій. Далі розглянемо їх більш ретельно.

Cucumber підхід. Користувачу треба відкрити будь-який feature файл. Знаходяться вони за наступним шляхом: ‘src/test/recources/features”. Кожен такий файл підписаний відповідно до списку тестових сценаріїв, котрі він містить в собі (рис. 4.1).

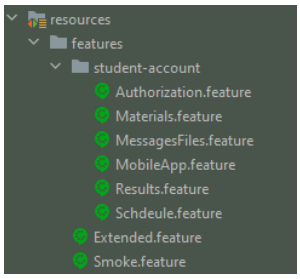


Рисунок 4.1 – Список feature файлів, котрі містять тестові сценарії

Наступним кроком буде налаштування конфігурації. Для цього треба відкрити відповідне меню (рис. 4.2).

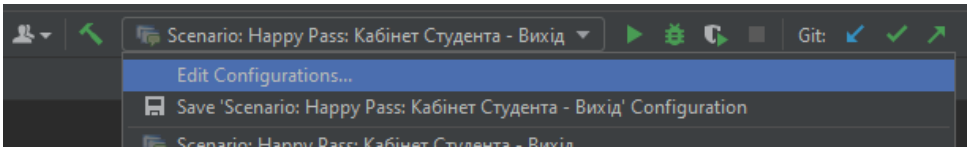


Рисунок 4.2 – Меню конфігурації Cucumber ранеру

У вікні “Run/Debug Configuration” в поле “Environments variables” ввести значення “LOCAL_RUN=true” (рис. 4.3).

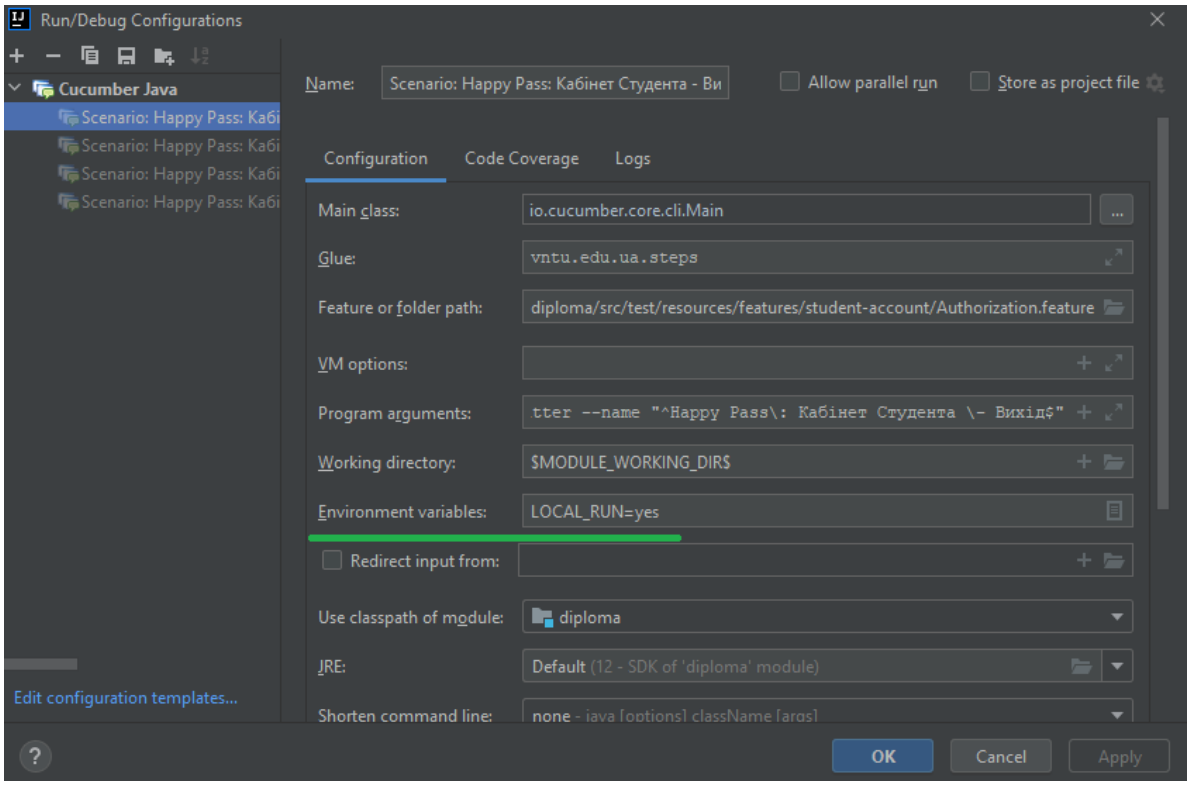


Рисунок 4.3 – Вікно Run/Debug Configuration

Конфігурацію для запуску тесту на основі Cucumber технології зроблено, далі користувачу треба запустити сценарій. Для цього у feature файлі, який він відкрив попередньо, треба натиснути на невеликий зелений трикутник (рис. 4.4). Після цього автоматичний тест запуститься, і внизу відкриється діалогове вікно, яке містить інформацію про хід тестування.

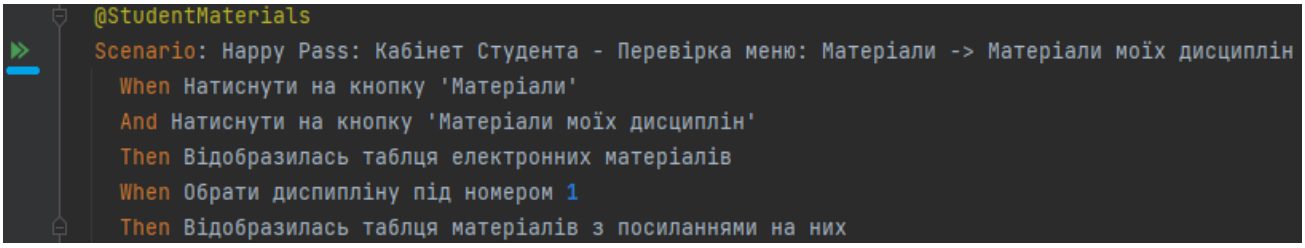


Рисунок 4.4 – Кнопка запуску тестового сценарію у feature файлі

JUnit підхід. Користувачу треба відкрити Runner.java файл, котрий знаходиться за шляхом: “src/test/java/runner”. Потім в поле “tags” необхідно ввести тег обраного сценарію, котрий користувач хоче запустити (рис. 4.5). Приклад тегу сценарію показано на рис. 4.4 (жовтим кольором). Конфігурацію треба налаштувати по прикладу Cucumber підходу (рис. 4.2 та 4.3). Потім необхідно запустити ранер, натиснувши на кнопку у вигляді зеленого трикутника (рис. 4.5).

Рисунок 4.5 – Приклад JUnit ранер

Тепер розглянемо детально віддалений варіант тестування, який реалізований уза допомогою GitLab CI/CD.

Для початку користувачу потрібно перейти за посиланням на віддалений репозиторій GitLab та відкрити меню зліва CI/CD – Pipelines. Перед цим йому необхідно запросити доступ до віддаленого репозиторію, оскільки було вирішено закрити доступ звичайним користувачам для запуску пайплайну в цілях безпеки. Далі на цій сторінці треба натиснути на кнопку “Run Pipeline”, котра розташована у правому верхньому кутку екрану. На новій сторінці необхіжно ще раз натиснути на синю кнопку “Run Pipeline”. Додаткові тестові параметри вводити не потрібно (але при необхідності можна).

Якщо користувач зробив усе правильно, то повинна була з'явитись сторінка, яка зображена на рис. 4.6.

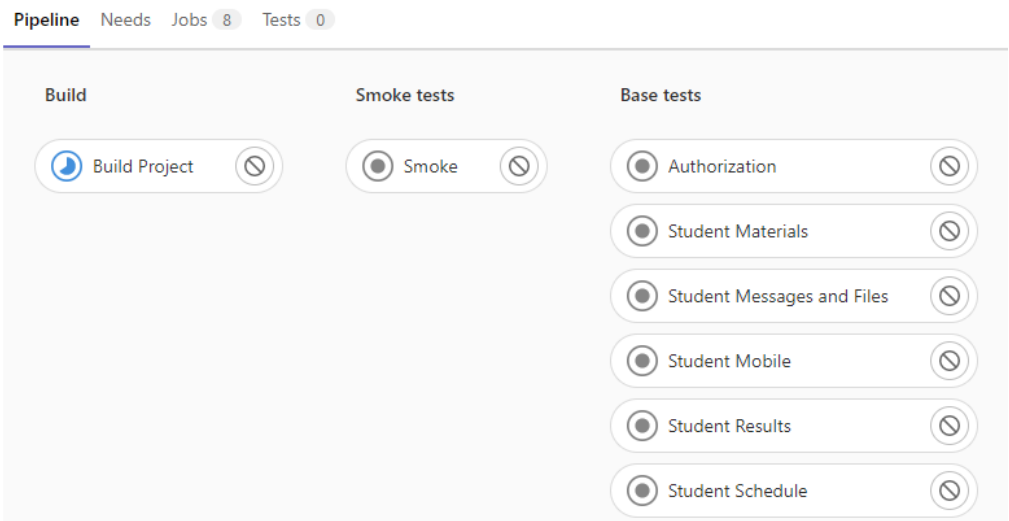


Рисунок 4.6 – Сторінка CI/CD пайплайну

На рис. 4.6 бачимо, що пайплайн успішно стартував. Якщо Build Project джоба виявиться червоною після її виконання, то її слід відкрити та подивитись логи, можливо користувач допустив помилку у конфігурації. У випадку, коли Build Project джоба виявиться зеленою, користувач все зробив правильно і CI/CD пайлайн стартував коректно і готовий до проведення тестування.

4.2 Проведення локального тестування

Щоб провести локальне тестування було обрано Cucumber підхід. Технічні характеристики локальної машини наведено в табл. 4.1.

Таблиця 4.1 – Технічні характеристики локальної машини

ЦП	Intel i5-12400F
Об’єм оперативної пам’яті	32 GB DDR4
Об’єм жорсткого диску	2 ТБ (вільно 42%)
Графічний пристрій	AMD Radeon RX5700XT 8GB
Операційна система	Windows 11 64-bit

Результати тестування занесено до табл. 4.2. Список тест-кейсів наведено у розділі 2.

Таблиця 4.2 – Результати проведеного локального тестування

Тест-кейс	Статус	Час (секунди)
Т-1	Пройдено успішно	8.4
Т-2	Пройдено успішно	8.8
Т-3	Пройдено успішно	8.5
Т-4	Пройдено успішно	8.5
Т-5	Пройдено успішно	9.2
Т-6	Пройдено успішно	9.4
Т-7	Пройдено успішно	9.5
Т-8	Пройдено успішно	8.4
Т-9	Пройдено успішно	8.6
Т-10	Пройдено успішно	8.8
Т-11	Пройдено успішно	11.5
Т-12	Пройдено успішно	9.0

Приклад проходження сценаріїв наведено на рис. 4.7.

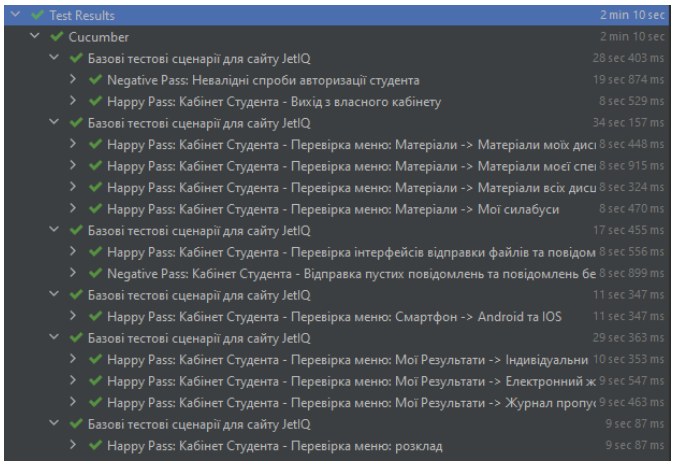


Рисунок 4.7 – Приклад проходження автотесту

Слід зазначити, що кожен feature файл було запущено окремо один від одного. Цез роблено для того, щоб краще було аналізувати результати тестування. В результаті усі тестові сценарії є пройденими успішно. Загальний час проходження базових тестів склав 108.6 секунд, а середній – 9.1. Оскільки, кожен тест успішно завершено, немає необхідності створювати нових багів.

4.3 Проведення тестування за допомогою CI/CD пайплайну

Для того щоб провести тестування на віддаленій машині скористаємось інструкціює користувача, яка була наведена у розділі 4.1. Віддалена віртуальна машина базується на ОС Linux.

Слід також зауважити, що пайплайн містить вісім джоб, серед яких шість джоб, котрі містять автоматизовані тест-кейси. Для зручності тестові сценарії розподілені по джобам за допомогою тегів. Кожна джоба має власну навзу, котра відповідає набору тестів, які до неї входять.

На рис. 4.8 зображено логи, які містять результати роботи однієї із джоб.

```
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 49.262 s
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:11 min
[INFO] Finished at: 2022-04-09T16:44:09Z
[INFO] ------------------------------------------------------------------------
Cleaning up project directory and file based variables

Job succeeded
```

Рисунок 4.8 – Логи з результатами роботи джоби

Результати тестування зображано на рис. 4.9 та занесено до табл. 4.3. Список тест-кейсів наведено у розділі 2.2.

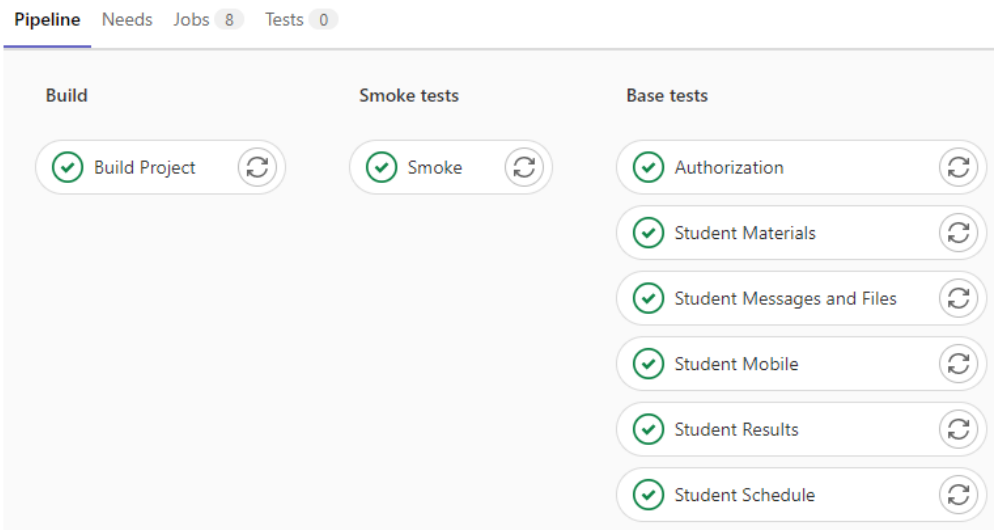


Рисунок 4.9 – Проходження тестування на віддаленій віртуальній машині

Таблиця 4.3 – Результати проведення тестування на віддаленій машині

Тест-кейси	Статус	Час (с)
Т-1 Т-2 Т-3 Т-4	Пройдено успішно	71.4
Т-5 Т-6 Т-7	Пройдено успішно	68.2
Т-8	Пройдено успішно	31.5
Т-9 Т-10	Пройдено успішно	50.3
Т-11	Пройдено успішно	43.1
Т-12	Пройдено успішно	40.6

Таким чином, ми бачимо, що всі тестові сценарії були виконані. Загальний час проходження базових тестів становив 305.1 секунд, а середній час проходження становив 25.4. Оскільки всі тести завершилися, можна вважати, що тестування завершилося успішно, і не потрібно створювати нових багів.

Крім того, слід зазначити, що середній і загальний час тестування зросли. Це пояснюється тим, що перед початком набору тестових сценаріїв у кожній джобі відбувається зборка проєкту, оскільки кожна джоба є окремою одна від одної та є по суті окремим середовищем.

4.4 Висновки

Четвертий розділ містить детальну інструкцію користувача. За допомогою цих інструкцій будь-який користувач може запустити тести на власній локальній або віддаленій машині.

Крім того, було проведено комплексне тестування системи JetIQ на власній локальній машині та за допомогою пайплайну CI/CD. У процесі тестування всі сценарії були протестовані, і не було виявлено жодних нових багів.

Автоматизований фреймворк показав, що здатен проводити регресійне, системне та інтеграційне тестування системи керування навчальним процесом JetIQ.

5. АНАЛІЗ ЕКОНОМІЧНИХ ПОКАЗНИКІВ РОЗРОБКИ

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Автоматизація тестування системи керування навчальним процесом JetIQ» відноситься до науково-технічних робіт, які плануються для використання безпосередньо самим розробником (замовником), тобто її результатами буде користуватися тільки одна особа – розробник (або замовник). У цьому випадку нам потрібно довести ефективність інвестицій, вкладених у цей проект самим розробником (замовником).

Для цього випадку необхідно виконати такі етапи робіт:

1) провести технологічний аудит власної науково-технічної розробки, тобто встановити її науково-технічний рівень;

2) розрахувати витрати на здійснення науково-технічної розробки;

3) провести розрахунок економічної ефективності науково-технічної розробки у випадку її впровадження розробником (замовником) на власному підприємстві та обґрунтувати економічну доцільність впровадження розробником (замовником) розробленого науково-технічного проекту.

Метою проведення комерційного і технологічного аудиту дослідження за темою «Автоматизація тестування системи керування навчальним процесом JetIQ» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [51].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1.

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	1	1	1
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	32	34	33
Середньоарифметична сума балів $СБ_c$	33,0		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизація тестування системи керування навчальним процесом JetIQ» становить 33,0 бала, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [52]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i ,$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{аі}},$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного *і*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{аі}}{I_{ні}};$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i,$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{ai}},$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного i-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}};$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 5.4.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Час генерації тестових даних	Год	0,5	0,1	5	0,2
Час виконання тестів на локальній машині	Год	2	0,2	10	0,25
Час виконання тестів на віддаленій машині	Год	2,5	0,3	8,33	0,2
Час генерації звітів	Год	1	0,2	5	0,25
Підтримувана база даних	ГБ	1000	2000	2	0,1

Узагальнений коефіцієнт якості (Вн) для нового технічного рішення складе:

$5 \cdot 0,2+10 \cdot 0,25+8,33 \cdot 0,2+5 \cdot 0,25+2 \cdot 0,1 = 6,62.$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 6,62 рази.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Автоматизація тестування системи керування навчальним процесом JetIQ», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p},$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 12800{,}00 \cdot 42 / 21 = 25600{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-технічної роботи	12800,00	609,52	42	25600,00
Інженер-розробник автоматизованих систем	11200,00	533,33	40	21333,33
Всього				46933,33

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Автоматизація тестування системи керування навчальним процесом JetIQ» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i ,$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} ,$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}35 / (21 \cdot 8) = 59{,}22$ *грн.*

$З_{р1} = 59{,}22 \cdot 8{,}00 = 473{,}79$ *грн.*

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця розробника автоматизованиї системи	8,00	2	1,10	59,22	473,79
Всього					473,79

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%},$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (46933{,}33 + 473{,}79) \cdot 11 / 100\% = 5214{,}78$ *грн.*

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (46933,33 + 473,79 + 5214,78) · 22 / 100% = 11576,82 грн.

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Автоматизація тестування системи керування навчальним процесом JetIQ».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj},$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1$ = 3 · 210,00 · 1,05 - 0 · 0 = 661,50 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір А4 500 аркушів клас-С Crystal Print&Copy UP	210,00	3	0	0	661,50

Продовження таблиці 5.7

Папір офісний Офіс Центр А5 80г/м2 500 аркушів клас С	129,00	3	0	0	406,35
Прибор настільний 13 предметів 6300-01 Buromax чорний	220,00	2	0	0	462,00
Набір канцелярський офісний FAX	210,00	2	0	0	441,00
Картридж для принтера	1100,00	1	0	0	1155,00
Диск оптичний CD-RW	25,00	5	0	0	131,25
USB флеш накопичувач Transcend 64Gb JetFlash 700 (TS64GJF700)	279,00	1	0	0	292,95
Всього					3550,05

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Автоматизація тестування системи керування навчальним процесом JetIQ», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 1 \cdot 3250,00 \cdot 1,05 = 3412,50$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Графічний адаптер: - тип відеокарти Інтегрована - відеокарта Vega 8 - об'єм відеопам'яті 2 ГБ	1	3250,00	3412,50
Всього			3412,50

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення. Витрати за статтею відсутні.

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ ,$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 2150,00 \cdot 1 \cdot 1,01 = 2171,50$ грн.

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Тестовий фреймворк	1	2150,00	2171,50
ОС Windows	1	4299,00	4341,99
Прикладний пакет Microsoft Office	1	5240,00	5292,40
Прикладний пакет моделювання процесів MatLab	1	5411,00	5465,11
CI/CD пайплайн	1	1850,00	1868,50
Всього			19139,50

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12},$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (16499{,}00 \cdot 2) / (2 \cdot 12) = 1374{,}92$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Локальна машина (ПК) Ноутбук ASUS Vivobook 15 X1500EA-BQ3733 (90NB0TY6-M040W0) Transparent Silver / 15.6" IPS Full HD / Intel Core i3-1115G4 / RAM 12 ГБ / SSD 512 ГБ	16499,00	2	2	1374,92
Обчислювальний комплекс та комп'ютеризована система проектування Комп'ютер ARTLINE D31 (D31v05)	59999,00	2	2	4999,92
Робоче місце розробника автоматичної системи	6520,00	5	2	217,33
Пристрій графічного виводу інформації	6750,00	4	2	281,25
Офісна оргтехніка	7300,00	4	2	304,17
Приміщення лабораторії	286000,00	35	2	1361,90
Мережеве обладнання	6899,00	5	2	229,97
Всього				8769,45

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i},$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; приймемо $Ц_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,05 \cdot 320,0 \cdot 7,50 \cdot 0,95 / 0,97 = 120,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Локальна машина (ПК) Ноутбук ASUS Vivobook 15 X1500EA-BQ3733 (90NB0TY6-M040W0)	0,05	320,0	120,00
Обчислювальний комплекс та комп'ютеризована система проектування Комп'ютер ARTLINE D31 (D31v05)	0,42	320,0	1008,00
Робоче місце розробника автоматичної системи	0,08	320,0	192,00
Пристрій графічного виводу інформації	0,22	4,2	6,93
Офісна оргтехніка	0,45	1,5	5,06
Мережеве обладнання	0,05	300,0	112,50
Всього			1444,49

До статті «Службові відрядження» дослідної роботи на тему «Автоматизація тестування системи керування навчальним процесом JetIQ» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» відсутні.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (46933,33 + 473,79) · 50 / 100% = 23703,56 грн.

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (46933,33 + 473,79) · 100 / 100% = 47407,12 грн.

Витрати на проведення науково-дослідної роботи на тему «Автоматизація тестування системи керування навчальним процесом JetIQ» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.16)$$

$В_{заг}$= 46933,33 +473,79 +5214,78 +11576,82 +3550,05 +3412,50 +0,00 + 19139,50 + 8769,45 +1444,49 +0,00 +0,00 +23703,56 +47407,12 = 171625,40 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta},$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 171625,40 / 0,95 = 180658,31 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки від її впровадження

При виконанні даної роботи за темою «Автоматизація тестування системи керування навчальним процесом JetIQ» розглядається ситуація, коли замовник певної науково-технічної розробки використовує її тільки на своєму підприємстві (чи в організації) і не виводить її на ринок. У цьому випадку позитивним результатом від впровадження цієї науково-технічної розробки може бути

покращення певних економічних та фінансових показників діяльності підприємства.

Для визначення величини майбутнього економічного ефекту та ефективності розробки визначимо певні характеристики організації (ВНТУ).

Таблиця 5.12 – Вихідні дані замовника

Показник	Рік розробки	1-й рік	2-й рік	3-й рік	4-й рік
Чисельність працівників, які виконують визначені функції вручну, осіб	4	-	-	-	-
Середня заробітна плата працівника, який виконує відповідну функцію вручну, грн	8900,00	-	-	-	-
Приблизні витрати на розробку автоматизованої системи управління, грн	180658,31	-	-	-	-
Економія чисельності працівників виконання виробничої чи управлінської функції яких було автоматизовано у році що аналізується, осіб	-	2	2	0	0
Кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, шт	15000	-	-	-	-
Прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, у році що аналізується (відносно року до впровадження даної розробки), шт	-	9000	6000	0	0

В даному випадку майбутній економічний ефект та ефективність буде формуватися на основі використання таких показників: $\Delta П_я$ – зростання прибутку підприємства внаслідок зниження витрат на оплату праці працівників, які виконують окремі виробничі чи інформаційно-технічні управлінські функції, грн. Причому $\Delta П_я$ може бути визначено як:

$$\Delta П_я = \frac{ЧП \cdot ЗП \cdot 12}{N} - \frac{(0,2...0,6) \cdot ЗВ}{\Delta N_i},$$

де *ЧП* – чисельність працівників, які виконують визначені функції вручну, приймемо 4 осіб; *ЗП* – середня заробітна плата працівника, який виконує відповідну функцію вручну, приймемо 8900,00 грн (провідний фахівець); *ЗВ* – приблизні витрати на розробку автоматизованої системи управління, приймемо 180658,31 грн; *N* – кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, 15000 шт; ΔN_i – прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, у році що аналізується (відносно року до впровадження даної розробки), шт.

Зростання прибутку підприємства в 1-й рік впровадження розробки

$\Delta П_я$ =4·8900,00·12/15000 - 0,4·180658,31/9000 = 20,45 грн/функц.

Зростання прибутку підприємства в 2-й рік впровадження розробки

$\Delta П_я$ =4·8900,00·12/15000 - 0,4·180658,31/15000 = 23,66 грн/функц.

Зростання прибутку підприємства в 3-й рік впровадження розробки

$\Delta П_я$ =4·8900,00·12/15000 - 0,4·180658,31/15000 = 23,66 грн/функц.

Зростання прибутку підприємства в 4-й рік впровадження розробки

$\Delta П_я$ =4·8900,00·12/15000 - 0,4·180658,31/15000 = 23,66 грн/функц.

$П_я$ – прибуток, який отримує підприємство від автоматизації виконання окремої виробничої чи інформаційно-технічної управлінської функції у кожному із років після впровадження науково-технічної розробки, грн. Даний прибуток можна приблизно оцінити виходячи з формули:

$$П_я = \frac{\Delta ЧП \cdot ЗП \cdot 12}{N},$$

де $\Delta ЧП$ – економія чисельності працівників виконання виробничої чи управлінської функції яких було автоматизовано у році що аналізується, осіб;

Прибуток який отримує підприємство від автоматизації функції в 1-й рік

$П_я$=2·8900,00·12/15000 = 14,24 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 2-й рік

$П_я$=2·8900,00·12/15000 = 14,24 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 3-й рік

$П_я$=0·8900,00·12/15000 = 0 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 4-й рік

$П_я$=0·8900,00·12/15000 = 0 грн/функц.

Збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\Delta П_я \cdot N + П_я \cdot \Delta N)_i,$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження на підприємстві результатів науково-технічної розробки у році що аналізується;

N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів нової науково-технічної розробки;

$П_я$ – основний якісний показник, який визначає результати діяльності підприємства у кожному із років після впровадження науково-технічної розробки;

ΔN – зміна основного кількісного показника діяльності підприємства в результаті впровадження науково-технічної розробки у році що аналізується.

Збільшення чистого прибутку підприємства в 1-й рік впровадження

$\Delta П_i$=20,45·15000+14,24·9000=434921,13 грн.

Збільшення чистого прибутку підприємства в 2-й рік впровадження

$\Delta П_i$=23,66·15000+14,24·(9000+6000)=568536,68 грн.

Збільшення чистого прибутку підприємства в 3-й рік впровадження

$\Delta П_i$=23,66·15000+0·(9000+6000+0)=354936,68 грн.

Збільшення чистого прибутку підприємства в 4-й рік впровадження

$\Delta П_i$=23,66·15000+0·(9000+6000+0+0)=354936,68 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати замовник від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t},$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,13;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання замовником додаткових чистих прибутків у цьому році.

$ПП$ =434921,13/(1+0,13)^1+568536,68/(1+0,13)^2+354936,68/(1+0,13)^3+

+354936,68/(1+0,13)^4=384885,95+445247,61+245988,92+217689,31= 1293811,79

грн.

Величина початкових інвестицій PV , які замовник має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ,$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=1,05;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 180658,31 грн.

$PV = k_{інв} \cdot ЗВ = 1{,}05 \cdot 180658{,}31 = 189691{,}23$ грн.

Абсолютний економічний ефект $E_{абс}$ для розробника від можливого впровадження науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1293811,79 грн;

PV – теперішня вартість початкових інвестицій, 189691,23 грн.

$E_{абс} = ПП - PV = 1293811{,}79 - 189691{,}23 = 1104120{,}57$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 1104120,57 грн;

PV – теперішня вартість початкових інвестицій, 189691,23 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1104120{,}57/189691{,}23)^{1/4} = 0{,}616.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,616 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Автоматизація тестування системи керування навчальним процесом JetIQ» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в},$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,616 = 1,62 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника профінансувати впровадження даної розробки для застосування в діяльності підприємства.

5.5 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизація тестування системи керування навчальним процесом JetIQ» становить 33,0 бала, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 6,62 рази.

Також термін окупності становить 1,62 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника до впровадження даної розробки при отриманні ефекту в розмірі 1104120,57 грн.

Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Автоматизація тестування системи керування навчальним процесом JetIQ».

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено тестовий фреймворк на базі технології Selenium WebDriver, який призначений автоматизувати тестування для системи керування навчальним процесом JetIQ.

Досліджено особливості розробки тестового фреймворку для автоматизації тестування. Проаналізовано ряд існуючих інструментів та технологій, які призначені для цього. В результаті ретельного аналізу вирішено розробляти тестовий фреймворк на основі технології Selenium WebDriver із застосуванням BDD методології.

Визначено перелік вимог, які необхідно протестувати. Створено 12 тест-кейсів. На основі вимог і тест-кейсів створено матрицю трасування. Створено загальну архітектуру та алгоритм роботи програми. Архітектура базується на технології Selenium, патерні Page Object та базових принципах ООП. Також створено загальну схему CD/CD пайплайну і в ході планування вирішено підключити проєкт до Git системи.

У результаті розробки тестового фреймворку створено автоматизовані тестові сценарії за допомогою Cucumber інструменту українською мовою. Налаштовано Git та GitLab CI/CD пайплайн для тестування на віддаленій машині у віртуальному середовищі.

Наведено чітку покрокову інструкцію користувача. Також проведено загальне тестування системи JetIQ на локальній машині та за допомогою CI/CD пайплайну. В ході тестування усі сценарії були пройдені, нових багів виявлено не було.

На основі ретельного та детального аналізу економічних показників розробки дійшли висновку про доцільність проведення науково-дослідної роботи за темою «Автоматизація тестування системи керування навчальним процесом JetIQ».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Автоматизоване тестування. URL: https://studopedia.com.ua/1_151546_avtomatizovane-testuvannya.html (дата звернення 10.10.2023).

2) Muszka Peter (2019) The ART of Automation: Leverage for companies, managers and testers. - Independently published. 150 p.

3) Автоматизоване тестування. URL: https://uk.wikipedia.org/wiki/Автоматизоване_тестування (дата звернення 10.10.2023).

4) Mr Navneesh Garg (2014) Test Automation using Selenium WebDriver with Java: Step by Step Guide. - Independently published. 404 p.

5) Boni Garcia (2022) Hands-On Selenium WebDriver with Java: A Deep Dive into the Development of End-to-End Tests. - Independently published. 419 p.

6) Інструменти автоматизації тестування. URL: https://uk.education-wiki.com/9205738-automation-testing-tools (дата звернення 10.10.2023)

7) Mark Chatham (2013) Selenium By Example - Volume I: Selenium IDE. - Lulu.com. 168 p.

8) Selenium. URL: https://uk.wikipedia.org/wiki/Selenium (дата звернення 10.10.2023).

9) Bandana Ojha (2018) Selenium Testing Interview Q & A: Selenium Testing Tool Paperback. - Independently published. 64 p.

10) Kochiaro Kevin (2018) Selenium Framework Design in DDT. - Packt Publishing. 102 p.

11) Selenium. URL: https://promdevelop.com/technologies/selenium/ (дата звернення 11.10.2023).

12) Автоматизуємо тестування: коли, навіщо і кому це потрібно. URL: https://newline.tech/test-automation-when-why-and-who-needs-it_uk/ (дата звернення 11.10.2023).

13) Micro Focus Unified Functional Testing. URL: https://en.wikipedia.org/wiki/Micro_Focus_Unif_Functional_Testing (дата звернення 11.10.2023).

14) Корнієнко М. М. (2008) Системи управління базами даних. Microsoft Access: Теоретичні основи, приклади та завдання, практичні роботи. - Ранок. 50 с.

15) Chaminda Chandrasekara (2019) Hands-On Functional Test Automation. - Apress. 267 p.

16) Короткий огляд мови програмування C#. URL: https://docs.microsoft.com/ua-ua/dotnet/csharp/tour-of-csharp/ (дата звернення 11.10.2023).

17) Visual Studio. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/quickstart-ide-orientation?view=vs-2019 (дата звернення 11.10.2023).

18) Apache JMeter. URL: https://jmeter.apache.org/ (дата звернення 11.10.2023).

19) Postman. URL: https://www.postman.com/ (дата звернення 11.10.2023).

20) Тест-кейси. URL: https://training.qatestlab.com/blog/course-materials/preconditions-test-cases/ (дата звернення 11.10.2023).

21) Матрица трасування. URL: https://habr.com/ua/company/simbirsoft/blog/412677/ (дата звернення 11.10.2023).

22) Boby Jose (2021) Test Automation: A manager's guide. - BCS. 274 p.

23) Васюхно М. В. (2016) Віртуальна та доповнена реальності. - Diss. Сумський державний університет. 165 с.

24) Основні елементи архітектури тестового фреймворку. URL: https://github.com/COMAQA/Selenium-Webdriver-lectures/blob/master/-page_object_pattern_arhitektura (дата звернення 11.10.2023).

25) Anton Angelov (2021) Design Patterns for High-Quality Automated Tests: Clean Code for Bulletproof Tests. - Independently published. 322 p.

26) Anand Hooda (2020) Cucumber Interview Questions and Answers. - Independently published. 56 p.

27) Автоматизація тестування: як уникнути поширених помилок. URL: https://www.globallogic.com/ua/insights/blogs/qa-automation-2/ (дата звернення 12.10.2023).

28) Автоматизація тестування на великих проектах: для чого і як ми її здійснюємо. URL: https://evergreens.com.ua/ua/articles/automated-testing.html (дата звернення 12.10.2023).н

29) Lou Pedron (2015) Software Test Automation: Getting Started Guide for QA Managers, Quality Engineers and Project Managers. - Lou Pedron. 41 p.

30) Zhamak Dehghani (2023) Data Mesh: Delivering Data-Driven Value at Scale. - Ascent Audio. 74 p.

31) Життєвий цикл розробки програмного забезпечення за Agile-методологією. Що це таке і які фази охоплює? URL: https://training.epam.ua/ua/blog/581 (дата звернення 12.10.2023).

32) Pallavi Sharma (2022) Selenium with Java – A Beginner's Guide: Web Browser Automation for Testing using Selenium with Java. - BPB Publications. 268 p.

33) Що таке .NET і чим займаються .NET-розробники. URL: https://training.epam.ua/ua/blog/301 (дата звернення 12.10.2023).

34) Створення проєкту автоматизації та написання UI тестів. URL: https://qalight.ua/kursy/automation/stvorennya-proektu-avtomatizaczii-ta-napisannya-ui-testiv/ (дата звернення 12.10.2023).

35) Carl Cocchiaro (2018) Selenium Framework Design in Data-Driven Testing. - Packt Publishing. 356 p.

36) Joe Colantonio (2023) Automation Awesomeness: 260 actionable affirmations to improve your QA and automation testing skills. - Test Guild. 282 p.

37) Автоматизоване мобільне тестування: перші кроки. URL: https://www.globallogic.com/ua/insights/blogs/first-steps-in-mobile-test-automation/ (дата звернення 12.10.2023).

38) Тренди автоматизації тестування у 2022-му. URL: https://careers.easternpeak.com/blog/qa-automation-trends/ (дата звернення 12.10.2023).

39) Paul R. Daugherty (2018) Human + Machine: Reimagining Work in the Age of AI. - Audible Studios. 35 p.

40) Chhavi Raj Dosaj (2020) The Self-Taught Software Tester A Step By Step Guide to Learn Software Testing Using Real-Life Project. - Independently published. 217 p.

41) Як «продати» автотести: проводимо розрахунки та розбираємо на прикладі. URL:https://highload.today/uk/blogs/yak-prodati-avtotesti-provodimo-rozrahunki-ta-rozbirayemo-na-prikladi/ (дата звернення 14.10.2023).

42) John Sonmez (2020) Soft Skills: The Software Developer's Life Manual. - Simple Programmer. 503 p.

43) Як оцінити проєкт з автотестування. URL: https://qamania.org/blog/як-оцінити-проєкт-з-автотестування/ (дата звернення 14.10.2023).

44) 10 термінів з автоматизації тестування. URL: https://training.epam.ua/ua/blog/562?lang=ua (дата звернення 14.10.2023).

45) 4 принципи автоматизованого тестування для більшої надійності автотестів. URL: https://aw.club/global/uk/blog/work/4-automated-testing-principles (дата звернення 14.10.2023).

46) Kristin Jackvony (2021) The Complete Software Tester: Concepts, Skills, and Strategies for High-Quality Testing. - Independently published. 514 p.

47) Reelav Patel (2020) QA Analyst/Software Tester Job Interview Questions & Answers: Winning The QA/Software Testing Interview. - Independently published. 101 p.

48) Мій топ анти-патернів у автоматизації тестування. URL: https://olegzarevych.medium.com/мій-топ-анти-патернів-у-автоматизації-тестування-72785668c3cc (дата звернення 12.10.2023).

49) Як писати автотести з Selenium. URL: https://robotdreams.cc/uk/blog/247-kak-pisat-avtotesty-s-selenium (дата звернення 12.10.2023).

50) Liliana Iancu (2019) QA Quality Assurance & Software Testing Fundamentals. - Independently published. 244 p.

51) Козловський В. О., Лесько О. Й., Кавецький В. В. (2021) Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт.

52) Кавецький В. В. (2016) Економічне обґрунтування інноваційних рішень: практикум.

ДОДАТКИ

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: **«Автоматизація тестування системи керування навчальним процесом JetIQ»**

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96,6% Схожість 3,3%

Аналіз звіту подібності (відмітити потрібне)

☑ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Андрій СИМОН
(підпис) (прізвище, ініціали)

Керівник роботи _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Додаток Б. Технічне завдання

ВНТУ

ЗАТВЕРДЖЕНО
Т.в.о. зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____Марія ЮХИМЧУК
"06" 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Автоматизація тестування системи керування навчальним процесом JetIQ

(тема)

08-33.МКР.007.00.000 ТЗ

Студент групи 2АКІТ-22м

Андрій СИМОН
Ім'я ПРІЗВИЩЕ

Керівник професор каф. КСУ

Володимир ДУБОВОЙ
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Найменування та галузь застосування

1.1. Назва – Автоматизація тестування системи керування навчальним процесом JetIQ

1.2 Галузь застосування – навчальний процес у закладі ВНТУ.

2. Підстава для розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від

№247 від 18-09-2023р.

3. Мета та призначення розробки.

Метою даної роботи є підвищення ефективності системи підтримки навчального процесу JetIQ шляхом автоматизації її тестування.

4. Джерела розробки.

Основні літературні джерела, на основі яких буде виконуватись МКР:

1. Bandana O. M., 2018. Selenium Testing Interview Q&A. Kyiv.

2. Kochiaro A. S., 2018. Selenium Framework Design in DDT. Packt Publishing.

3. Hyda A. A., 2020. Cucumber Interview Questions and Answers. Kyiv: Lybid.

4. Pinakin A. B., 2018. Page Object Model using Selenium WebDriver & Java. Kyiv.

5. Технічні вимоги

5.1. Перелік головних функцій:

– оптимізація тестового процесу;

– запуск автотестів на локальній та віддаленій машині;

– автоматичний запуск автотестів по графіку;

– генерація та зберігання тестових результатів.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

– WINDOWS 10;

– Java JDK 11;

– IntellJ IDEA.

5.2.2. Умови експлуатації системи:

– робота на веб-додатках;

– можливість запуску автотестів у будь-який момент.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «03»_09__ 2023 р.
2. Удосконалення технології при автоматизації навчального процесу «10»__09___ 2023 р.
3. Визначення технічних характеристик системи «15»__09___ 2023 р.
4. Розробка програмного забезпечення системи «21»___09__ 2023 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «30»__09__ 2023 р.
2. Розробка моделі бази даних системи «05»__10___ 2023 р.

3. Тестування програмного забезпечення «10»__10___ 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28»__11_ 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «11»__12_ 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «20»__12_ 2023 р.

Додаток В. Лістинг програми

```
@Smoke
Feature: Тестові смок сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ

   Scenario: Happy Pass: Перевірка роботи домашньої сторінки
     Then Навігаційні посилання відображаються на сторінці
     And Розділи меню верхньої панелі відображаються на сторінці

  Scenario: Happy Pass: Вхід студента у власний кабінет
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'
    Then Студента звати 'Симон Андрій Дмитрович'


Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'

  @Authorization
  Scenario Outline: Negative Pass: Невалідні спроби авторизації студента
    When Ввести '<typeFirst>' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести '<typeSecond>' значення в поле паролю
    And Натиснути на кноку 'Ввійти'
    Then Вискочило повідомлення про неправильний логін або пароль

    Examples:
      | typeFirst | typeSecond |
      | invalid   | invalid    |
      | valid     | invalid    |
      | invalid   | valid      |

  @Authorization
  Scenario: Happy Pass: Кабінет Студента - Вихід з власного кабінету
    When Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'
    Then Студента звати 'Симон Андрій Дмитрович'
```

```
    When Натиснути на кноку 'Вихід'
    Then Навігаційні посилання відображаються на сторінці
    And Розділи меню верхньої панелі відображаються на сторінці

Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали ->
Матеріали моїх дисциплін
    When Натиснути на кнопку 'Матеріали'
    And Натиснути на кнопку 'Матеріали моїх дисциплін'
    Then Відобразилась таблця електронних матеріалів
    When Обрати диспипліну під номером 1
    Then Відобразилась таблця матеріалів з посиланнями на них

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали ->
Матеріали моєї спеціальності
    When Натиснути на кнопку 'Матеріали'
    And Натиснути на кнопку 'Матеріали моєї спеціальності'
    And Обрати будь який семестр
    Then Відобразилась таблця електронних матеріалів
    When Обрати диспипліну під номером 1
    Then Відобразилась таблця матеріалів з посиланнями на них

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали ->
Матеріали всіх дисциплін
    When Натиснути на кнопку 'Матеріали'
    And Натиснути на кнопку 'Матеріали всіх дисцпиплін'
    Then 'Електронні навчальні матеріали' table is appeared

  @StudentMaterials
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Матеріали -> Мої
силабуси
    When Натиснути на кнопку 'Мої силабуси'
    Then Відобразилась таблця матеріалів з посиланнями на них


Feature: Базові тестові сценарії для сайту JetIQ
```

```
  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'

  @StudentMessagesFiles
  Scenario: Happy Pass: Кабінет Студента - Перевірка інтерфейсів відправки
файлів та повідомлень викладачу
    When Натиснути на кнопку 'Повідомлення'
    And Натиснути на кнопку 'Відправити повідомлення'
    Then Відобразився інтферейс 'Повідомлення викладачу'
    When Повернутись на вкладку назад і натиснути на кнопку 'Надіслати
файл'
    Then Відобразився інтферейс 'Надіслати файл викладачу'

  @StudentMessagesFiles
  Scenario: Negative Pass: Кабінет Студента - Відправка пустих повідомлень
та повідомлень без файлів викладачу
    When Натиснути на кнопку 'Повідомлення'
    And Натиснути на кнопку 'Відправити повідомлення'
    Then Відобразився інтферейс 'Повідомлення викладачу'
    When Натиснути на кнопку 'Відправити'
    Then Відобразилось повідомлення про помилку відправки пустого
повідомлення
    When Повернутись на вкладку назад і натиснути на кнопку 'Надіслати
файл'
    Then Відобразився інтферейс 'Надіслати файл викладачу'
    When Натиснути на кнопку 'Відправити'
    Then Відобразився попап із помилкою відправки файла

Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'

  @StudentMobile
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Смартфон ->
Android та IOS
    When Навести на кнопку 'Смартфон'
    And Натиснути на кнопку 'Андроїд'
    Then Відкрилась сторінка додатка у Play Market
```

```
    When Повернутись назад та натиснути на кнопку 'IOS'
    Then Відкрилась сторінка додатка у Apple Store

Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'
    When Натиснути на кноку 'Мої Результати'
    Then Відкрилась сторінка із результатами тестів і посиланнями

  @StudentResults
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Мої Результати -
> Індивідуальний план студента
    When Натиснути на кноку 'Індивідуальний план студента'
    Then Відкрилась сторінка із індивідуальним навчальним планом студента

  @StudentResults
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Мої Результати -
> Електронний журнал успішності
    When Натиснути на кноку 'Електронний журнал успішності'
    Then Відкрилась сторінка із електронним журналом успішності

  @StudentResults
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: Мої Результати -
> Журнал пропусків та заборгованостей
    When Натиснути на кноку 'Журнал пропусків'
    Then Відкрилась сторінка із журналом пропусків
    When Повернутись назад і натиснути на кноку 'Журнал заборгованостей'
    Then Відкрилась сторінка із журналом заборгованостей

Feature: Базові тестові сценарії для сайту JetIQ

  Background:
    Given Відкрити домашню сторінку JetIQ
    And Натиснути на кноку 'Вхід'
    And Ввести 'valid' значення в поле логіну
    And Натиснути на кнопку 'Далі'
    And Ввести 'valid' значення в поле паролю
    And Натиснути на кноку 'Ввійти'

  @StudentSchedule
  Scenario: Happy Pass: Кабінет Студента - Перевірка меню: розклад
    When Натиснути на кнопку 'Розклад'
    Then Відобразилась таблиця 'Розклад'
```

```
package vntu.edu.ua.runner;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

import static vntu.edu.ua.utils.Constants.FEATURES;
import static vntu.edu.ua.utils.Constants.GLUE;

@RunWith(Cucumber.class)
@CucumberOptions(
        glue = GLUE,
        features = FEATURES,
        plugin = {"pretty", "html:target/cucumber"})
public class Runner { }

image: markhobson/maven-chrome:latest

stages:
  - Build
  - Smoke Tests
  - Base Tests
  - Extended Tests

.run_cucumber_tests: &run_cucumber_tests |
  mvn -f ${CI_PROJECT_DIR}/pom.xml \
  -Dcucumber.filter.tags="${TEST_TAG}" test

Build Project:
  stage: Build
  allow_failure: false
  script:
    - mvn -f ${CI_PROJECT_DIR}/pom.xml -Dmaven.test.skip=true clean package

Smoke:
  variables:
    TEST_TAG: "@Smoke"
  stage: Smoke Tests
  allow_failure: false
  script:
    - *run_cucumber_tests

Student Materials:
  variables:
    TEST_TAG: "@StudentMaterials and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests
```

```
Student Schedule:
  variables:
    TEST_TAG: "@StudentSchedule and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests

Student Results:
  variables:
    TEST_TAG: "@StudentResults and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests

Student Messages and Files:
  variables:
    TEST_TAG: "@StudentMessagesFiles and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests

Student Mobile:
  variables:
    TEST_TAG: "@StudentMobile and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests

Authorization:
  variables:
    TEST_TAG: "@Authorization and not @Disabled"
  stage: Base Tests
  allow_failure: true
  script:
    - *run_cucumber_tests

<?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>vntu.edu.ua</groupId>
```

```
<artifactId>diploma</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
  <java.version>11</java.version>
  <encoding>UTF-8</encoding>

  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
  <maven-compiler-plugin.version>3.7.0</maven-compiler-plugin.version>
  <maven.surefire.version>2.22.1</maven.surefire.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
  </dependency>
  <dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>6.8.1</version>
  </dependency>
  <dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>6.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
  </dependency>
  <dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>4.3.1</version>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
```

```
      <version>1.2.3</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven-compiler-plugin.version}</version>
        <configuration>
          <source>${java.version}</source>
          <target>${java.version}</target>
          <encoding>${encoding}</encoding>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>${maven.surefire.version}</version>
        <configuration>
          <argLine>-Xmx32768m</argLine>
          <includes>
            <include>**/Runner.java</include>
          </includes>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

```
package vntu.edu.ua.manager;

import lombok.AllArgsConstructor;
import org.openqa.selenium.WebDriver;
import vntu.edu.ua.pages.*;

@AllArgsConstructor
public class PageFactoryManager {

    private final WebDriver webDriver;

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

    public SignInPage getSignInPage() {
        return new SignInPage(webDriver);
```

```
    }

    public StudentHomePage getUserHomePage() {
        return new StudentHomePage(webDriver);
    }

    public MaterialsPage getMaterialsPage() {
        return new MaterialsPage(webDriver);
    }

    public MessagesFilesPage getMessagesFilesPage() {
        return new MessagesFilesPage(webDriver);
    }

    public MyResultsPage getMyResultsPage() {
        return new MyResultsPage(webDriver);
    }
}
```

```
package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.util.ArrayList;
import java.util.List;

import static vntu.edu.ua.utils.Constants.*;

public abstract class BasePage {

    @Getter
    protected WebDriver webDriver;

    public BasePage(WebDriver webDriver) {
        this.webDriver = webDriver;
        PageFactory.initElements(webDriver, this);
    }

    public String getActiveTab(int tab) {
        List<String> tabs = new ArrayList<>(webDriver.getWindowHandles());
        return tabs.get(tab);
    }
```

```
    public void waitForPageLoadComplete() {
        getWebDriverWait().until(driver -> ((JavascriptExecutor) driver)
                .executeScript(PAGE_LOAD_SCRIPT).equals(COMPLETE));
    }

    public void waitForElementVisibility(WebElement webElement) {

getWebDriverWait().until(ExpectedConditions.visibilityOf(webElement));
    }

    public void waitForElementClickable(WebElement webElement) {

getWebDriverWait().until(ExpectedConditions.elementToBeClickable(webElement
));
    }

    public boolean isElementDisplayed(WebElement webElement) {
        return webElement.isDisplayed();
    }

    private WebDriverWait getWebDriverWait() {
        return new WebDriverWait(webDriver, DEFAULT_TIMEOUT);
    }
}

package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

import java.util.List;

import static vntu.edu.ua.utils.Constants.HOME_PAGE_URL;

@Getter
public class HomePage extends BasePage {

    @FindBy(xpath =
"//nav[@id='mainav']//a[@href='https://my.vntu.edu.ua/user/']")
    private WebElement sighInButton;

    @FindBy(xpath = "//nav[@id='mainav']/ul/li")
    private List<WebElement> navigationLinks;

    @FindBy(xpath = "//div[@class='exampl1']")
    private List<WebElement> topBarMenuSections;
```

```
    public HomePage(WebDriver webDriver) {
        super(webDriver);
    }

    public void openHomePage() {
        webDriver.get(HOME_PAGE_URL);
    }

    public void clickOnSighInButton() {
        waitForElementClickable(sighInButton);
        sighInButton.click();
    }
}
```

```
package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

import java.util.List;

@Getter
public class MaterialsPage extends BasePage {


    @FindBy(xpath = "//h4/a[@href='/method/subj2.php']")
    private WebElement myDisciplinesMaterialsButton;

    @FindBy(xpath = "//h4/a[@href='/method/sem2.php']")
    private WebElement mySpecialtyMaterialsButton;

    @FindBy(xpath = "//h4/a[@href='/method/index2.php']")
    private WebElement allMaterialsButton;

    @FindBy(xpath = "//tr//td/a[not(contains(@href, 'teacher'))]")
    private List<WebElement> subjectsInMaterialTable;

    @FindBy(xpath = "//table//th[text()='Код.']")
    private WebElement teacherMaterialsTable;

    @FindBy(xpath = "//a[text()='4 курс 8 триместр ']")
    private WebElement searchBySemesterMaterials;

    @FindBy(xpath = "//table[@class='table table-hover table-condensed']")
    private WebElement electronicMaterialsTable;

    @FindBy(xpath = "//h4[contains(text(), 'Ефективність')]")
```

```
    private WebElement allMaterialsTablePage;

    public MaterialsPage(WebDriver webDriver) {
        super(webDriver);
    }

    public void clickMyDisciplinesMaterialsButton() {
        waitForElementClickable(myDisciplinesMaterialsButton);
        myDisciplinesMaterialsButton.click();
    }

    public void clickOnSubjectsInMaterialTable(int value) {
        WebElement subject = subjectsInMaterialTable.get(value - 1);
        waitForElementVisibility(subject);
        waitForElementClickable(subject);
        subject.click();
    }

    public void clickOnMySpecialtyMaterialsButton() {
        waitForElementClickable(mySpecialtyMaterialsButton);
        mySpecialtyMaterialsButton.click();
    }

    public void clickOnSearchBySemesterMaterials() {
        waitForElementClickable(searchBySemesterMaterials);
        searchBySemesterMaterials.click();
    }

    public void clickOnAllMaterialsButton() {
        waitForElementClickable(allMaterialsButton);
        allMaterialsButton.click();
    }
}

package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

@Getter
public class MessagesFilesPage extends BasePage {

    @FindBy(xpath = "//a[@class='bd-linkbutton-2  bd-button-13  bd-own-
margins bd-content-element']")
    private WebElement sendMessageButton;
```

```
    @FindBy(xpath = "//h1[text()='Повідомлення викладачу у ПК і на
електронну пошту.']")
    private WebElement messageToProfPage;

    @FindBy(xpath = "//a[@class='bd-linkbutton-5  bd-button-52  bd-own-
margins bd-content-element']")
    private WebElement sendFileButton;

    @FindBy(xpath = "//form[@enctype='multipart/form-data']")
    private WebElement sendFilePage;

    @FindBy(xpath = "//input[@value='Надіслати']")
    private WebElement sendMessageToProfButton;

    @FindBy(xpath = "//font[contains(text(), 'Відсутній текст')]")
    private WebElement messageIsEmptyError;

    public MessagesFilesPage(WebDriver webDriver) {
        super(webDriver);
    }

    public void clickOnSendMessageButton() {
        waitForElementClickable(sendMessageButton);
        sendMessageButton.click();
    }

    public void clickOnSendFileButton() {
        webDriver.switchTo().window(getActiveTab(0));
        waitForElementClickable(sendFileButton);
        sendFileButton.click();
    }

    public void clickOnSendMessageToProfButton() {
        waitForElementClickable(sendMessageToProfButton);
        sendMessageToProfButton.click();
    }
}

package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

import java.util.List;

@Getter
public class MyResultsPage extends BasePage {
```

```
@FindBy(xpath = "//th[text()='Назва тесту']")
private WebElement tableWithTestResults;

@FindBy(xpath = "//a[text()='Індивідуальний план студента']")
private WebElement studentPlanButton;

@FindBy(xpath = "//th[text()='Дисципліна']")
private List<WebElement> studentPlanPage;

@FindBy(xpath = "//a[text()='Електронний журнал успішності']")
private WebElement logOfRateButton;

@FindBy(xpath = "//td[text()='Модулі']")
private List<WebElement> logOfRatesTable;

@FindBy(xpath = "//a[text()='Журнал пропусків']")
private WebElement passLogButton;

@FindBy(xpath = "//td[text()='Дисципліна']")
private WebElement passLogTable;

@FindBy(xpath = "//a[text()='Журнал заборгованостей']")
private WebElement debtLogButton;

@FindBy(xpath = "//h1[contains(text(), 'Ceci')]")
private WebElement debtLogInfo;

public MyResultsPage(WebDriver webDriver) {
    super(webDriver);
}

public void clickOnStudentPlanButton() {
    waitForElementClickable(studentPlanButton);
    studentPlanButton.click();
}

public void clickOnLogOfRateButton() {
    waitForElementClickable(logOfRateButton);
    logOfRateButton.click();
}

public void clickOnPassLogButton() {
    waitForElementClickable(passLogButton);
    passLogButton.click();
    webDriver.switchTo().window(getActiveTab(2));
}

public void clickOnDebtLogButton() {
```

```
        webDriver.switchTo().window(getActiveTab(1));
        waitForElementClickable(debtLogButton);
        debtLogButton.click();
        webDriver.switchTo().window(getActiveTab(3));
    }
}
package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

@Getter
public class SignInPage extends BasePage {

    @FindBy(xpath = "//input[@id='user_field']")
    private WebElement usernameField;

    @FindBy(xpath = "//input[@id='pwd_field']")
    private WebElement passwordField;

    @FindBy(xpath = "//div[@class='login-header']")
    private WebElement loginHeader;

    @FindBy(xpath = "//button[@id='do-verify-login']")
    private WebElement nextButton;

    @FindBy(xpath = "//button[@id='do-verify-password']")
    private WebElement enterButton;

    @FindBy(xpath = "//div[@class='alert alert-warning']")
    private WebElement authErrorMessage;

    public SignInPage(WebDriver webDriver) {
        super(webDriver);
    }

    public void setValueToUsernameField(String value) {
        waitForElementVisibility(usernameField);
        waitForElementClickable(usernameField);
        usernameField.sendKeys(value);
    }

    public void setValueToPasswordField(String value) {
        waitForElementVisibility(passwordField);
        waitForElementClickable(passwordField);
        passwordField.sendKeys(value);
    }
```

```
    public void clickOnLoginHeader() {
        waitForElementClickable(loginHeader);
        loginHeader.click();
    }

    public void clickOnNextButton() {
        waitForElementClickable(nextButton);
        nextButton.click();
    }

    public void clickOnEnterButton() {
        waitForElementClickable(enterButton);
        enterButton.click();
    }
}
```

```
package vntu.edu.ua.pages;

import lombok.Getter;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.FindBy;

@Getter
public class StudentHomePage extends BasePage {

    @FindBy(xpath = "//h2[contains(@class, 'bd-textblock-50')]")
    private WebElement studentName;

    @FindBy(xpath = "//a[@class='menufont' and text()='Матеріали']")
    private WebElement materialsMenu;
    @FindBy(xpath = "//a[text()='Мої силабуси']")
    private WebElement mySilabusesMenu;

    @FindBy(xpath = "//a[text()='Розклад']")
    private WebElement scheduleMenu;

    @FindBy(xpath = "//form[@name='myform']")
    private WebElement scheduleTable;

    @FindBy(xpath = "//a[text()='Мої результати']")
    private WebElement myResultsMenu;

    @FindBy(xpath = "//a[text()='Повідомлення']")
    private WebElement messagesMenu;

    @FindBy(xpath = "//a[text()='Смартфон']")
```

```
private WebElement mobileMenu;

@FindBy(xpath = "//a[text()='Android']")
private WebElement androidButton;

@FindBy(xpath = "//span[text()='JetIQ-Student']")
private WebElement playMarketPage;

@FindBy(xpath = "//a[text()='iOS 14+']")
private WebElement appleButton;

@FindBy(xpath = "//h1[text()='JetIQ-Student']")
private WebElement appleMarketPage;

@FindBy(xpath = "//a[text()='Вихід']")
private WebElement exitButton;

public StudentHomePage(WebDriver webDriver) {
    super(webDriver);
}

public void clickOnMaterialsMenu() {
    waitForElementClickable(materialsMenu);
    materialsMenu.click();
}

public void clickOnScheduleMenu() {
    waitForElementClickable(scheduleMenu);
    scheduleMenu.click();
    webDriver.switchTo().window(getActiveTab(1));
}

public void clickOnMySilabusesMenu() {
    Actions actions = new Actions(webDriver);
    actions.moveToElement(materialsMenu).build().perform();
    waitForElementClickable(mySilabusesMenu);
    mySilabusesMenu.click();
    webDriver.switchTo().window(getActiveTab(1));
}

public void clickOnMyResultsMenu() {
    waitForElementClickable(myResultsMenu);
    myResultsMenu.click();
    webDriver.switchTo().window(getActiveTab(1));
}

public void clickOnMessagesMenu() {
    waitForElementClickable(messagesMenu);
    messagesMenu.click();
```

```
    }

    public void clickOnMobileMenu() {
        waitForElementVisibility(mobileMenu);
        waitForElementClickable(mobileMenu);
        new Actions(webDriver).moveToElement(mobileMenu).build().perform();
    }

    public void clickOnAndroidButton() {
        waitForElementVisibility(androidButton);
        waitForElementClickable(androidButton);
        androidButton.click();
    }

    public void clickOnAppleButton() {
        webDriver.switchTo().window(getActiveTab(0));
        clickOnMobileMenu();
        waitForElementVisibility(appleButton);
        waitForElementClickable(appleButton);
        appleButton.click();
    }

    public void clickOnExitButton() {
        waitForElementClickable(exitButton);
        exitButton.click();
    }
}
```

```
package vntu.edu.ua.steps;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import lombok.extern.slf4j.Slf4j;
import org.openqa.selenium.WebElement;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

@Slf4j
public class AssertsSteps extends BaseSteps {

    @Then("Навігаційні посилання відображаються на сторінці")
    public void checkNavLinks() {
        log.info("Start checking navigation links...");
        for (WebElement webElement : homePage.getNavigationLinks()) {
            assertTrue(homePage.isElementDisplayed(webElement));
        }
        log.info("Navigation links are displayed on page");
```

```
    }

    @And("Розділи меню верхньої панелі відображаються на сторінці")
    public void checkTopBarMenuSections() {
        log.info("Start checking top bar menu sections...");
        for (WebElement webElement : homePage.getTopBarMenuSections()) {
            assertTrue(homePage.isElementDisplayed(webElement));
        }
        log.info("Top bar menu sections are displayed on page");
    }

    @Then("Студента звати {string}")
    public void checkStudentName(String studentName) {
        studentHomePage.waitForPageLoadComplete();

studentHomePage.waitForElementVisibility(studentHomePage.getStudentName());
        log.info("Start checking student name...");
        assertTrue(studentHomePage.getStudentName().getText().replace("\n",
" ").contains(studentName));
    }

    @Then("Вискочило повідомлення про неправильний логін або пароль")
    public void checkAuthErrorMessage() {

signInPage.waitForElementVisibility(signInPage.getAuthErrorMessage());
        log.info("Start checking error message...");

assertTrue(signInPage.getAuthErrorMessage().getText().contains("Неправильни
й"));
    }

    @Then("Відобразилась таблця матеріалів з посиланнями на них")
    public void checkMaterialLinks() {
        materialsPage.waitForPageLoadComplete();

materialsPage.waitForElementVisibility(materialsPage.getTeacherMaterialsTab
le());

assertTrue(materialsPage.isElementDisplayed(materialsPage.getTeacherMateria
lsTable()));
    }

    @Then("Відобразилась таблця електронних матеріалів")
    public void checkElectronicMaterialsTable() {
        materialsPage.waitForPageLoadComplete();

materialsPage.waitForElementVisibility(materialsPage.getElectronicMaterials
Table());
```

```
assertTrue(materialsPage.isElementDisplayed(materialsPage.getElectronicMate
rialsTable()));
    }

    @Then("'Електронні навчальні матеріали' table is appeared")
    public void checkAllMaterialsTable() {
        materialsPage.waitForPageLoadComplete();

materialsPage.waitForElementVisibility(materialsPage.getAllMaterialsTablePa
ge());

assertTrue(materialsPage.isElementDisplayed(materialsPage.getAllMaterialsTa
blePage()));
    }

    @When("Відобразилась таблиця 'Розклад'")
    public void checkScheduleTable() {

studentHomePage.waitForElementClickable(studentHomePage.getScheduleTable())
;

assertTrue(studentHomePage.isElementDisplayed(studentHomePage.getScheduleTa
ble()));
    }

    @Then("Відобразився інтферейс 'Повідомлення викладачу'")
    public void checkSendMessageInterface() {

messagesFilesPage.getWebDriver().switchTo().window(messagesFilesPage.getAct
iveTab(1));
        messagesFilesPage.waitForPageLoadComplete();

messagesFilesPage.waitForElementVisibility(messagesFilesPage.getMessageToPr
ofPage());

assertTrue(messagesFilesPage.isElementDisplayed(messagesFilesPage.getMessag
eToProfPage()));
    }

    @Then("Відобразився інтферейс 'Надіслати файл викладачу'")
    public void checkSendFileInterface() {

messagesFilesPage.getWebDriver().switchTo().window(messagesFilesPage.getAct
iveTab(2));
        messagesFilesPage.waitForPageLoadComplete();

messagesFilesPage.waitForElementVisibility(messagesFilesPage.getSendFilePag
e());
```

```
assertTrue(messagesFilesPage.isElementDisplayed(messagesFilesPage.getSendFi
lePage()));
    }

    @Then("Відобразилось повідомлення про помилку відправки пустого
повідомлення")
    public void checkEmptyMessageError() {
        messagesFilesPage.waitForPageLoadComplete();

messagesFilesPage.waitForElementVisibility(messagesFilesPage.getMessageIsEm
ptyError());

assertTrue(messagesFilesPage.isElementDisplayed(messagesFilesPage.getMessag
eIsEmptyError()));
    }

    @Then("Відобразився попап із помилкою відправки файла")
    public void checkEmptyFileError() {
        String popupMessage =
messagesFilesPage.getWebDriver().switchTo().alert().getText();
        assertTrue(popupMessage.contains("Не заповнені всі поля"));
    }

    @Then("Відкрилась сторінка додатка у Play Market")
    public void checkPlayMarketPage() {

studentHomePage.getWebDriver().switchTo().window(studentHomePage.getActiveT
ab(1));
        studentHomePage.waitForPageLoadComplete();

studentHomePage.waitForElementVisibility(studentHomePage.getPlayMarketPage(
));

assertTrue(studentHomePage.isElementDisplayed(studentHomePage.getPlayMarket
Page()));
    }

    @Then("Відкрилась сторінка додатка у Apple Store")
    public void checkAppleStorePage() {

studentHomePage.getWebDriver().switchTo().window(studentHomePage.getActiveT
ab(2));
        studentHomePage.waitForPageLoadComplete();

studentHomePage.waitForElementVisibility(studentHomePage.getAppleMarketPage
());
```

```
assertTrue(studentHomePage.isElementDisplayed(studentHomePage.getAppleMarke
tPage()));
    }

    @Then("Відкрилась сторінка із результатами тестів і посиланнями")
    public void checkTableWithTestResults() {
        myResultsPage.waitForPageLoadComplete();

myResultsPage.waitForElementVisibility(myResultsPage.getTableWithTestResult
s());

assertTrue(myResultsPage.isElementDisplayed(myResultsPage.getTableWithTestR
esults()));
    }

    @Then("Відкрилась сторінка із індивідуальним навчальним планом
студента")
    public void checkStudentPlanPage() {
        myResultsPage.waitForPageLoadComplete();

myResultsPage.waitForElementVisibility(myResultsPage.getStudentPlanPage().g
et(0));
        assertFalse(myResultsPage.getStudentPlanPage().isEmpty());
        for (WebElement webElement : myResultsPage.getStudentPlanPage()) {
            assertTrue(myResultsPage.isElementDisplayed(webElement));
        }
    }

    @Then("Відкрилась сторінка із електронним журналом успішності")
    public void checkLogOfRatesTable() {
        myResultsPage.waitForPageLoadComplete();

myResultsPage.waitForElementVisibility(myResultsPage.getLogOfRatesTable().g
et(0));
        assertFalse(myResultsPage.getLogOfRatesTable().isEmpty());
        for (WebElement webElement : myResultsPage.getLogOfRatesTable()) {
            assertTrue(myResultsPage.isElementDisplayed(webElement));
        }
    }

    @Then("Відкрилась сторінка із журналом пропусків")
    public void checkPassLogTable() {
        myResultsPage.waitForPageLoadComplete();

myResultsPage.waitForElementVisibility(myResultsPage.getPassLogTable());

assertTrue(myResultsPage.isElementDisplayed(myResultsPage.getPassLogTable()
));
```

```
    }

    @Then("Відкрилась сторінка із журналом заборгованостей")
    public void checkDebtLogInfo() {
        myResultsPage.waitForPageLoadComplete();

myResultsPage.waitForElementVisibility(myResultsPage.getDebtLogInfo());

assertTrue(myResultsPage.isElementDisplayed(myResultsPage.getDebtLogInfo())
);
    }
}

package vntu.edu.ua.steps;

import vntu.edu.ua.pages.*;

import static vntu.edu.ua.utils.Context.*;

public abstract class BaseSteps {
    protected HomePage homePage;
    protected SignInPage signInPage;
    protected StudentHomePage studentHomePage;
    protected MaterialsPage materialsPage;
    protected MessagesFilesPage messagesFilesPage;
    protected MyResultsPage myResultsPage;

    public BaseSteps() {
        homePage = (HomePage) scenarioContext.get(HOME_PAGE);
        signInPage = (SignInPage) scenarioContext.get(SIGN_IN_PAGE);
        studentHomePage = (StudentHomePage)
scenarioContext.get(USER_HOME_PAGE);
        materialsPage = (MaterialsPage)
scenarioContext.get(MATERIALS_PAGE);
        messagesFilesPage = (MessagesFilesPage)
scenarioContext.get(MESSAGES_FILES_PAGE);
        myResultsPage = (MyResultsPage)
scenarioContext.get(MY_RESULTS_PAGE);
    }
}

package vntu.edu.ua.steps;

import io.cucumber.java.en.And;
import io.cucumber.java.en.When;
import lombok.extern.slf4j.Slf4j;

@Slf4j
public class ClickSteps extends BaseSteps {
```

```
@When("Відкрити домашню сторінку JetIQ")
public void openHomePage() {
    homePage.openHomePage();
    homePage.waitForPageLoadComplete();
    log.info("Home page was opened");
}

@And("Натиснути на кноку 'Вхід'")
public void clickOnSighInButton() {
    homePage.clickOnSighInButton();
    log.info("Was clicked on 'Вхід' button");
}

@And("Натиснути на кнопку 'Далі'")
public void clickOnNextButton() {
    signInPage.clickOnLoginHeader();
    signInPage.clickOnNextButton();
    log.info("Was clicked on 'Далі' button");
}

@And("Натиснути на кноку 'Ввійти'")
public void clickOnEnterButton() {
    signInPage.clickOnEnterButton();
    log.info("Was clicked on 'Ввійти' button");
}

@When("Натиснути на кнопку 'Матеріали'")
public void clickOnMaterialsButton() {
    studentHomePage.waitForPageLoadComplete();
    studentHomePage.clickOnMaterialsMenu();
    log.info("Was clicked on 'Матеріали' button");
}

@And("Натиснути на кнопку 'Матеріали моїх дисциплін'")
public void clickOnMyDisciplineMaterialsButton() {
    materialsPage.clickMyDisciplinesMaterialsButton();
}

@When("Обрати диспипліну під номером {int}")
public void clickOnDisciplineNameButton(int value) {
    materialsPage.waitForPageLoadComplete();
    materialsPage.clickOnSubjectsInMaterialTable(value);
}

@When("Натиснути на кнопку 'Матеріали моєї спеціальності'")
public void clickOnMySpecialtyMaterialsButton() {
    materialsPage.clickOnMySpecialtyMaterialsButton();
}
```

```
    @When("Обрати будь який семестр")
    public void chooseAnySemester() {
        materialsPage.clickOnSearchBySemesterMaterials();
    }

    @When("Натиснути на кнопку 'Матеріали всіх дисцпиплін'")
    public void clickOnAllMaterialsButton() {
        materialsPage.clickOnAllMaterialsButton();
    }

    @When("Натиснути на кнопку 'Мої силабуси'")
    public void clickOnMySilabusesButton() {
        studentHomePage.clickOnMySilabusesMenu();
    }

    @When("Натиснути на кнопку 'Розклад'")
    public void clickOnScheduleButton() {
        studentHomePage.clickOnScheduleMenu();
    }

    @When("Натиснути на кнопку 'Повідомлення'")
    public void clickOnMessagesMenu() {
        studentHomePage.clickOnMessagesMenu();
    }

    @When("Натиснути на кнопку 'Відправити повідомлення'")
    public void clickOnSendMessageButton() {
        messagesFilesPage.clickOnSendMessageButton();
    }

    @When("Повернутись на вкладку назад і натиснути на кнопку 'Надіслати
файл'")
    public void clickOnSendFileButton() {
        messagesFilesPage.clickOnSendFileButton();
    }

    @When("Натиснути на кнопку 'Відправити'")
    public void clickOnSendButton() {
        messagesFilesPage.clickOnSendMessageToProfButton();
    }

    @When("Навести на кнопку 'Смартфон'")
    public void clickOnMobileMenu() {
        studentHomePage.clickOnMobileMenu();
    }

    @When("Натиснути на кнопку 'Андроїд'")
    public void clickOnAndroidButton() {
```

```
        studentHomePage.clickOnAndroidButton();
    }

    @When("Повернутись назад та натиснути на кнопку 'IOS'")
    public void clickOnAppleButton() {
        studentHomePage.clickOnAppleButton();
    }

    @When("Натиснути на кноку 'Мої Результати'")
    public void clickOnMyResultsMenu() {
        studentHomePage.clickOnMyResultsMenu();
    }

    @When("Натиснути на кноку 'Індивідуальний план студента'")
    public void clickOnStudentPlanButton() {
        myResultsPage.clickOnStudentPlanButton();
    }

    @When("Натиснути на кноку 'Електронний журнал успішності'")
    public void clickOnLogOfRateButton() {
        myResultsPage.clickOnLogOfRateButton();
    }

    @When("Натиснути на кноку 'Журнал пропусків'")
    public void clickOnPassLogButton() {
        myResultsPage.clickOnPassLogButton();
    }

    @When("Повернутись назад і натиснути на кноку 'Журнал
заборгованостей'")
    public void clickOnDebtLogButton() {
        myResultsPage.clickOnDebtLogButton();
    }

    @When("Натиснути на кноку 'Вихід'")
    public void clickOnExitButton() {
        studentHomePage.waitForPageLoadComplete();
        studentHomePage.clickOnExitButton();
    }
}

package vntu.edu.ua.steps;

import io.cucumber.java.en.And;
import io.cucumber.java.en.When;
import lombok.extern.slf4j.Slf4j;

import static vntu.edu.ua.utils.Constants.*;
```

```
@Slf4j
public class FillFieldsSteps extends BaseSteps {

    @When("Ввести {string} значення в поле логіну")
    public void setUsernameToField(String valueType) {
        String value;
        if (valueType.equals(VALID)) {
            value = USERNAME;
        } else {
            value = INVALID_VALUE;
        }
        signInPage.setValueToUsernameField(value);
        log.info("Username was set");
    }

    @And("Ввести {string} значення в поле паролю")
    public void setPasswordToField(String valueType) {
        String value;
        if (valueType.equals(VALID)) {
            value = PASSWORD;
        } else {
            value = INVALID_VALUE;
        }
        signInPage.setValueToPasswordField(value);
        log.info("Password was set");
    }
}
```

```
package vntu.edu.ua.steps;

import io.cucumber.java.After;
import io.cucumber.java.Before;
import io.cucumber.java.Scenario;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import vntu.edu.ua.manager.PageFactoryManager;

import java.io.File;
import java.time.OffsetDateTime;

import static io.github.bonigarcia.wdm.WebDriverManager.chromedriver;
import static vntu.edu.ua.utils.Constants.*;
import static vntu.edu.ua.utils.Context.*;
```

```
@Slf4j
public class Hooks {

    private WebDriver webDriver;

    @Before(order = 1)
    public void setUpWebDriver() {
        chromedriver().setup();
        if (System.getenv(LOCAL_RUN) == null) {
            ChromeOptions chromeOptions = new ChromeOptions();
            chromeOptions.addArguments(NO_SANDBOX_OPTION);
            chromeOptions.addArguments(DISABLE_SETUID_OPTION);
            chromeOptions.addArguments(DISABLE_GPU_OPTION);
            chromeOptions.addArguments(HEADLESS_OPTION);
            chromeOptions.addArguments(DISABLE_DEV_SHM_OPTION);
            chromeOptions.addArguments(WINDOW_RESOLUTION_OPTION);
            chromeOptions.setBinary(GOOGLE_CHROME_LINUX_PATH);
            webDriver = new ChromeDriver(chromeOptions);
        } else {
            webDriver = new ChromeDriver();
            webDriver.manage().window().maximize();
        }
        scenarioContext.put(WEB_DRIVER, webDriver);

        log.info("WebDriver was started successfully");
    }

    @Before(order = 2)
    public void setUpPageFactory() {
        PageFactoryManager pageFactoryManager = new
PageFactoryManager(webDriver);
        scenarioContext.put(HOME_PAGE, pageFactoryManager.getHomePage());
        scenarioContext.put(SIGN_IN_PAGE,
pageFactoryManager.getSignInPage());
        scenarioContext.put(USER_HOME_PAGE,
pageFactoryManager.getUserHomePage());
        scenarioContext.put(MATERIALS_PAGE,
pageFactoryManager.getMaterialsPage());
        scenarioContext.put(MESSAGES_FILES_PAGE,
pageFactoryManager.getMessagesFilesPage());
        scenarioContext.put(MY_RESULTS_PAGE,
pageFactoryManager.getMyResultsPage());
        log.info("PageFactory was started successfully");
    }

    @SneakyThrows
    @After(order = 2)
    public void scenarioReporter(Scenario scenario) {
```

```
        if (scenario.isFailed()) {
            String fileName = OffsetDateTime.now().toEpochSecond() +
".png";
            log.info("Scenario is failed. Start making a screenshot...");
            File screenshot = ((TakesScreenshot)
webDriver).getScreenshotAs(OutputType.FILE);
            FileUtils.copyFile(screenshot, new File("target/generated-test-
sources/" + fileName));
        }
    }

    @After(order = 1)
    public void closeWebDriver() {
        webDriver.quit();
        log.info("WebDriver was closed successfully");
    }
}

package vntu.edu.ua.utils;

import static vntu.edu.ua.utils.Convertor.getDecodeValue;

public class Constants {

    public final static String HOME_PAGE_URL =
"https://jetiq.vntu.edu.ua/";

    public final static long DEFAULT_TIMEOUT = 60;
    public final static String PAGE_LOAD_SCRIPT = "return
document.readyState";
    public final static String COMPLETE = "complete";
    public final static String VALID = "valid";
    public final static String INVALID_VALUE = "INVALID_VALUE";
    public final static String LOCAL_RUN = "LOCAL_RUN";
    public final static String GLUE = "vntu.edu.ua.steps";
    public final static String FEATURES = "src/test/resources/features";

    public final static String USERNAME = getDecodeValue("MDQtMTgtMzk2");
    public final static String PASSWORD =
getDecodeValue("Q3VtZGVvMTg3Mg==");

    public final static String GOOGLE_CHROME_LINUX_PATH = "/usr/bin/google-
chrome";
    public final static String NO_SANDBOX_OPTION = "--no-sandbox";
    public final static String DISABLE_SETUID_OPTION = "--disable-setuid-
sandbox";
    public final static String DISABLE_GPU_OPTION = "--disable-gpu";
    public final static String HEADLESS_OPTION = "headless";
```

```
    public final static String DISABLE_DEV_SHM_OPTION = "--disable-dev-shm-
usage";
    public final static String WINDOW_RESOLUTION_OPTION = "--window-
size=1920x1080";
}

package vntu.edu.ua.utils;

import lombok.extern.slf4j.Slf4j;

import java.util.HashMap;

@Slf4j
public class Context {
    public static final HashMap<String, Object> scenarioContext = new
HashMap<>();

    public static final String WEB_DRIVER = "WEB_DRIVER";

    public static final String HOME_PAGE = "HOME_PAGE";
    public static final String SIGN_IN_PAGE = "SIGN_IN_PAGE";
    public static final String USER_HOME_PAGE = "USER_HOME_PAGE";
    public static final String MATERIALS_PAGE = "MATERIALS_PAGE";
    public static final String MESSAGES_FILES_PAGE = "MESSAGES_FILES_PAGE";
    public static final String MY_RESULTS_PAGE = "MY_RESULTS_PAGE";
}

package vntu.edu.ua.utils;

import java.util.Base64;

public class Convertor {

    public static String getDecodeValue(String key) {
        return new String(Base64.getDecoder().decode(key));
    }
}
```

Додаток Г. Ілюстративна частина

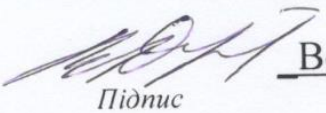
ІЛЮСТРАТИВНА ЧАСТИНА

АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ СИСТЕМИ КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ JETIQ

Студент групи _2АКІТ-22м_____

Підпис _Андрій СИМОН_
Ім'я ПРІЗВИЩЕ

Керівник _д.т.н., професор_____



Підпис _Володимир ДУБОВОЙ_
Ім'я ПРІЗВИЩЕ

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

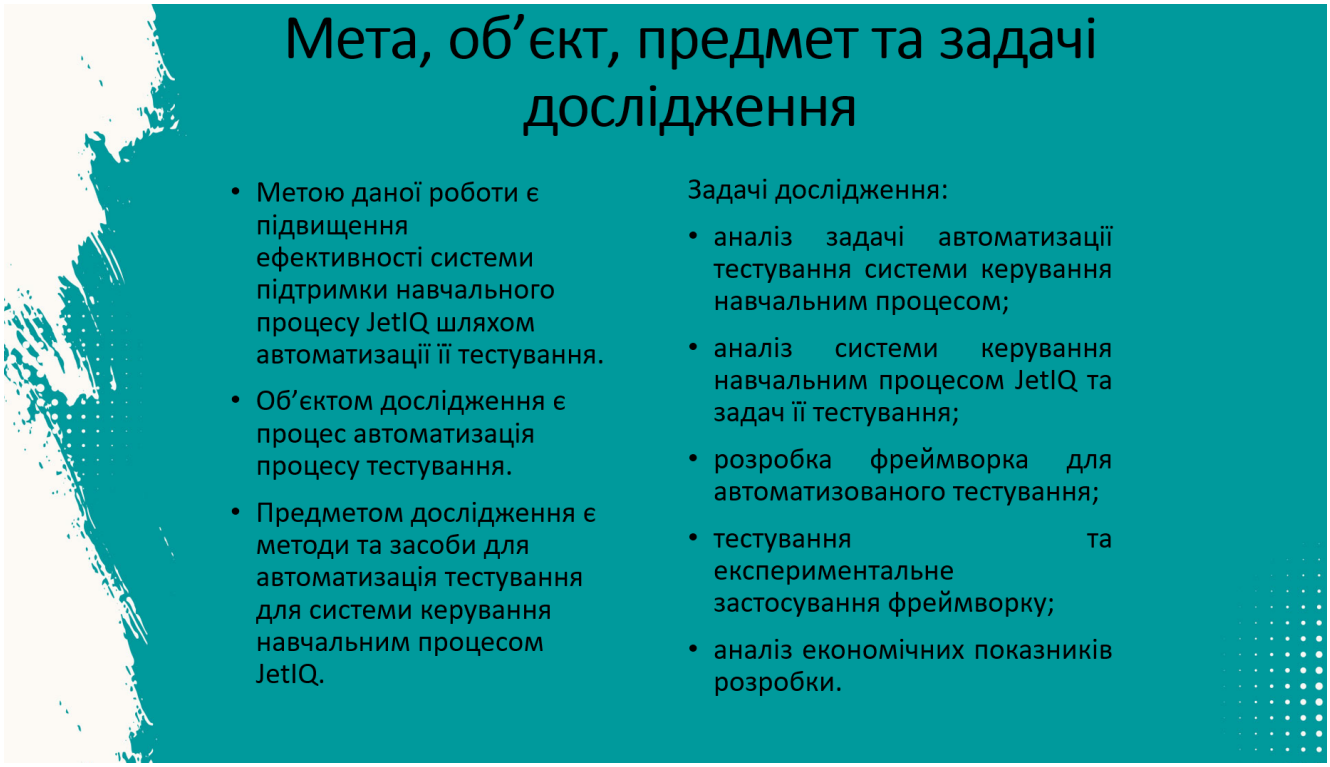


Рисунок Г.3 – Мета, об’єкт, предмет та задачі дослідження

Рисунок Г.4 – Новизна та практична цінність одержаних результатів

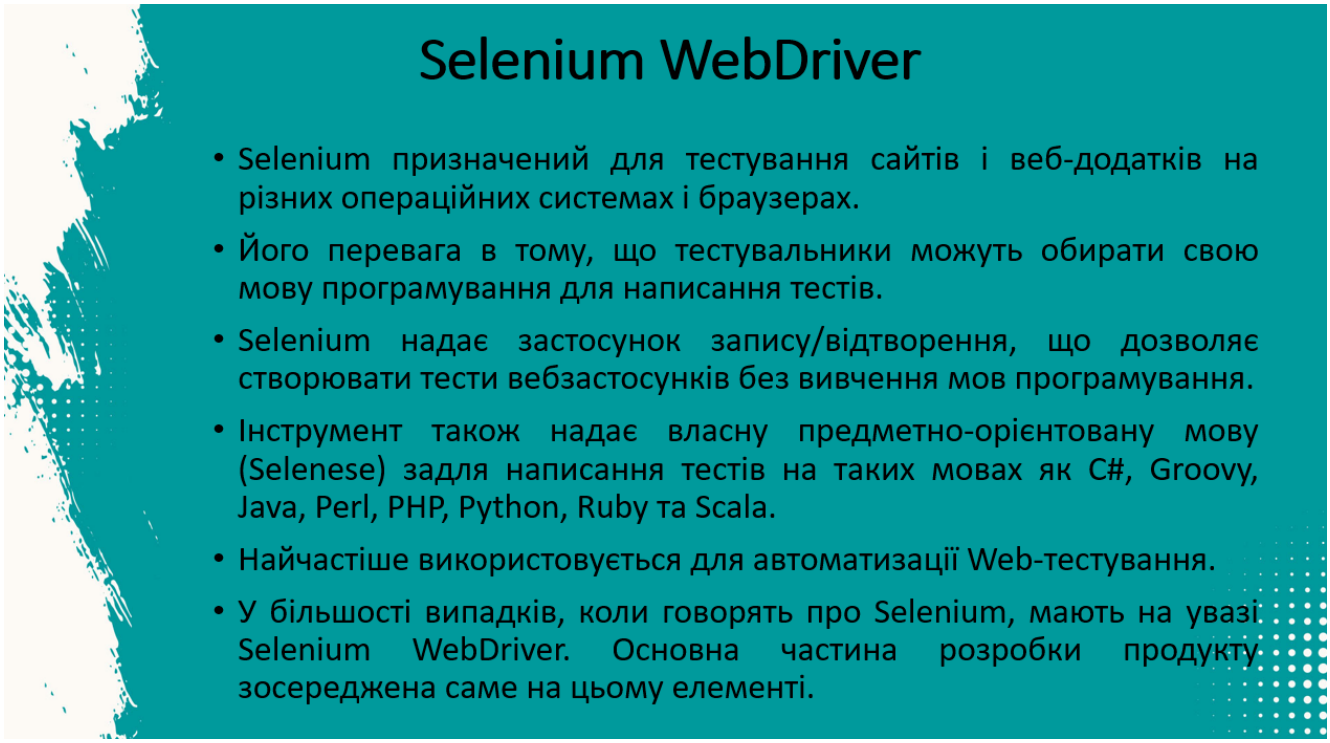


Рисунок Г.5 – Selenium WebDriver

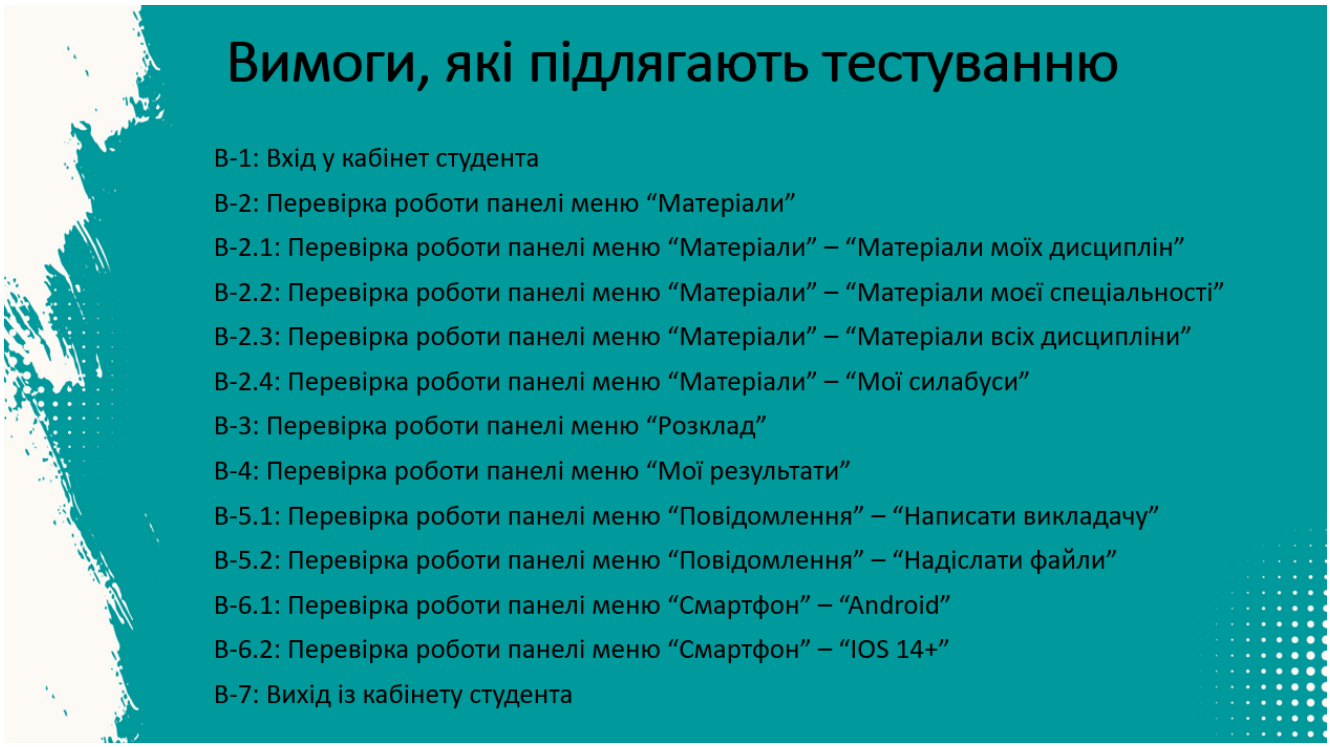


Рисунок Г.6 – Вимоги, які підлягають тестуванню

Список тест-кейсів

ID	Назва	Передумови	Кроки	Очікуючий результат
T-1	Кабінет Студента - Перевірка меню: Матеріали -> Матеріали моїх дисциплін	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Натиснути на кнопку 'Матеріали'	1. Натиснути на кнопку 'Матеріали моїх дисциплін'	Відобразилась таблиця електронних матеріалів
T-2	Кабінет Студента - Перевірка меню: Матеріали -> Матеріали моєї спеціальності		1. Натиснути на кнопку 'Матеріали моєї спеціальності'	
T-3	Кабінет Студента - Перевірка меню: Матеріали -> Матеріали всіх дисциплін		1. Натиснути на кнопку 'Матеріали всіх дисциплін'	Відобразилась таблиця "Електронні навчальні матеріали"
T-4	Кабінет Студента - Перевірка меню: Матеріали -> Мої силабуси	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента	1. Натиснути на кнопку 'Мої силабуси'	Відобразилась таблця матеріалів з посиланнями на них
T-5	Кабінет Студента - Перевірка меню: Мої Результати -> Індивідуальний план студента	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Натиснути на кнопку 'Мої Результати'	1. Натиснути на кнопку 'Індивідуальний план студента'	Відкрилась сторінка із індивідуальним навчальним планом студента
T-6	Кабінет Студента - Перевірка меню: Мої Результати -> Електронний журнал успішності		1. Натиснути на кнопку 'Електронний журнал успішності'	Відкрилась сторінка із електронним журналом успішності
T-7	Кабінет Студента - Перевірка меню: Мої Результати -> Журнал пропусків та заборгованостей		1. Натиснути на кнопку 'Журнал пропусків' 2. Повернутись назад і натиснути на кнопку 'Журнал заборгованостей'	1. Відкрилась сторінка із журналом пропусків 2. Відкрилась сторінка із журналом заборгованостей
T-8	Кабінет Студента - Вихід з власного кабінету	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента	1. Натиснути на кнопку 'Вихід'	Студент вийшов з власного кабінету

ID	Назва	Передумови	Кроки	Очікуючий результат
T-9	Кабінет Студента - Перевірка інтерфейсів відправки файлів та повідомлень викладачу	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Натиснути на кнопку 'Повідомлення'	1. Натиснути на кнопку 'Відправити повідомлення' 2. Повернутись на вкладку назад і натиснути на кнопку 'Надіслати файл'	1. Відобразився інтферейс 'Повідомлення викладачу' 2. Відобразився інтферейс 'Надіслати файл викладачу'
T-10	Кабінет Студента - Відправка пустих повідомлень та повідомлень без файлів викладачу		1. Натиснути на кнопку 'Відправити повідомлення' 2. Натиснути на кнопку 'Відправити' 3. Повернутись на вкладку назад і натиснути на кнопку 'Надіслати файл' 4. Натиснути на кнопку 'Відправити'	1. Відобразився інтферейс 'Повідомлення викладачу' 2. Відобразилось повідомлення про помилку відправки пустого повідомлення 3. Відобразився інтферейс 'Надіслати файл викладачу' 4. Відобразився попап із помилкою відправки файла
T-11	Кабінет Студента - Перевірка меню: Смартфон -> Android та IOS	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента 3. Навести на кнопку 'Смартфон'	1. Натиснути на кнопку 'Андроїд' 2. Повернутись назад та натиснути на кнопку 'IOS'	1. Відкрилась сторінка додатка у Play Market 2. Відкрилась сторінка додатка у Apple Store
T-12	Кабінет Студента - Перевірка меню: розклад	1. Відкрити сторінку сайту JetIQ 2. Увійти в кабінет студента	1. Натиснути на кнопку 'Розклад'	Відобразилась таблиця 'Розклад'

Рисунок Г.7 – Список тест-кейсів

Матриця трасування

ID	T-1	T-2	T-3	T-4	T-5	T-6	T-7	T-8	T-9	T-10	T-11	T-12
B-1	+	+	+	+	+	+	+	+	+	+	+	+
B-2	+	+	+	+								
B-2.1	+											
B-2.2		+										
B-2.3			+									
B-2.4				+								
B-3												+
B-4					+	+	+					
B-5.1									+	+		
B-5.2									+	+		
B-6.1											+	
B-6.2											+	
B-7								+				

Рисунок Г.8 – Матриця трасування

Рисунок Г.9 – Архітектура фреймворку

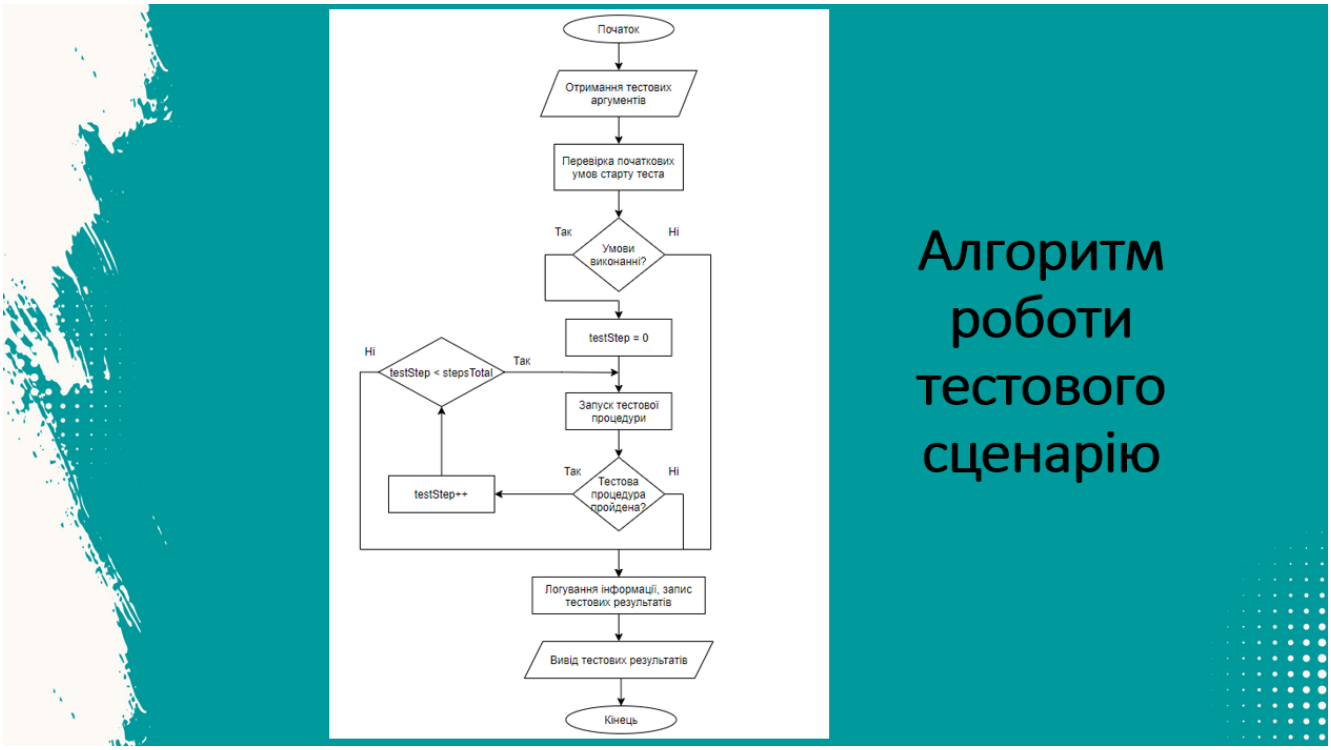


Рисунок Г.10 – Алгоритм роботи тестового сценарію

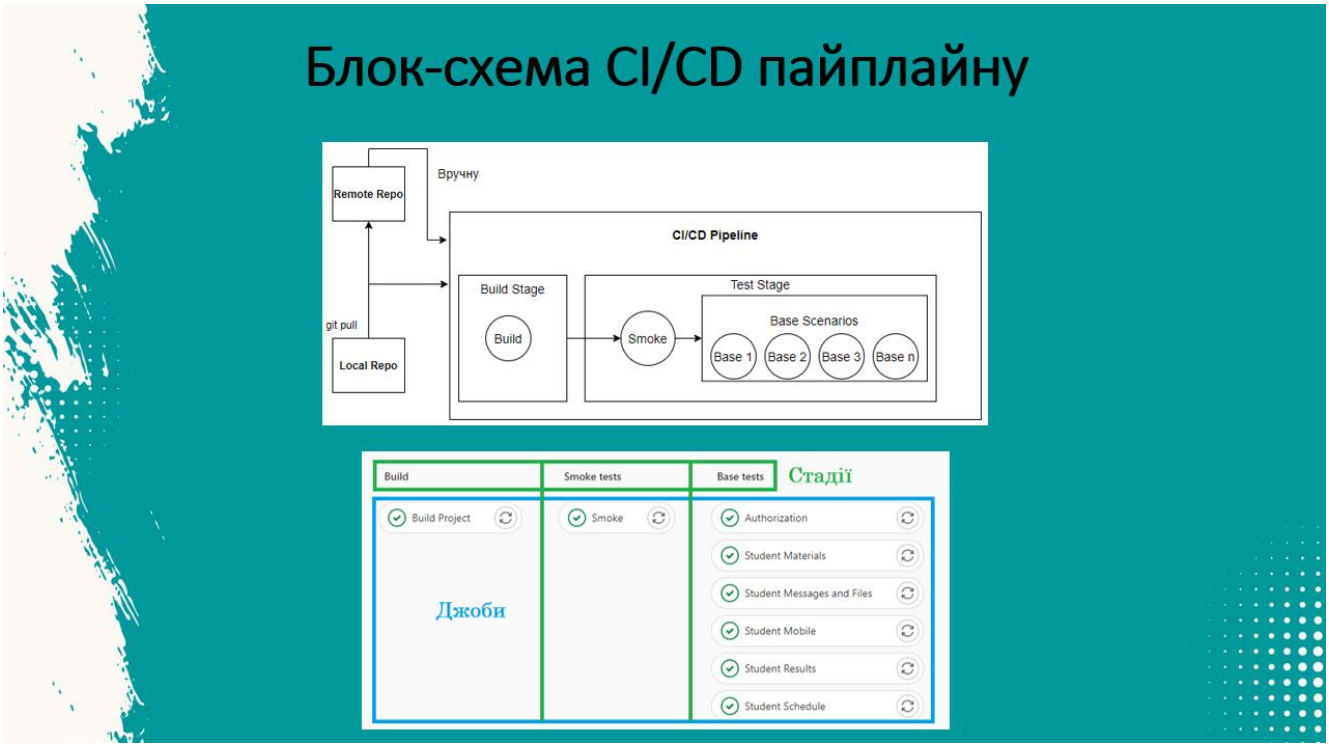


Рисунок Г.11 – Блок-схема CI/CD пайплайну

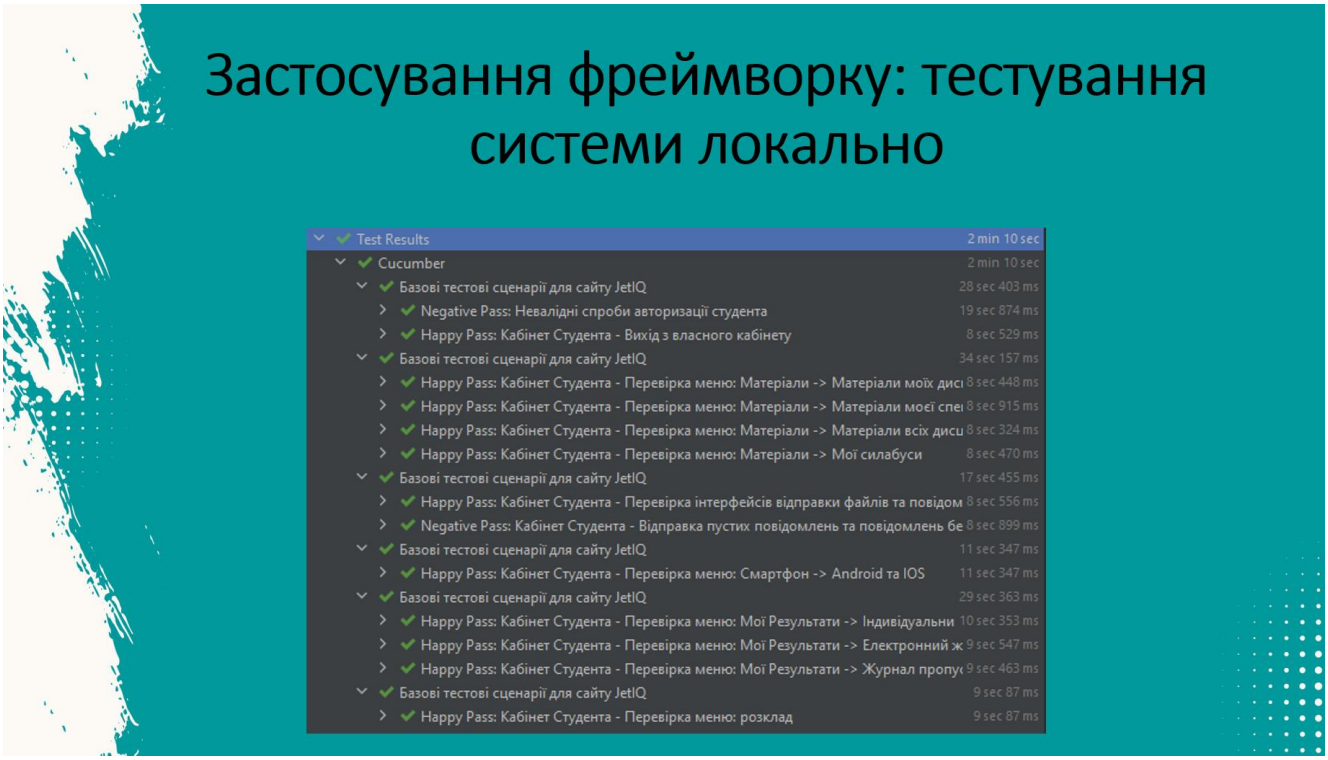


Рисунок Г.12 – Застосування фреймворку: тестування системи локально

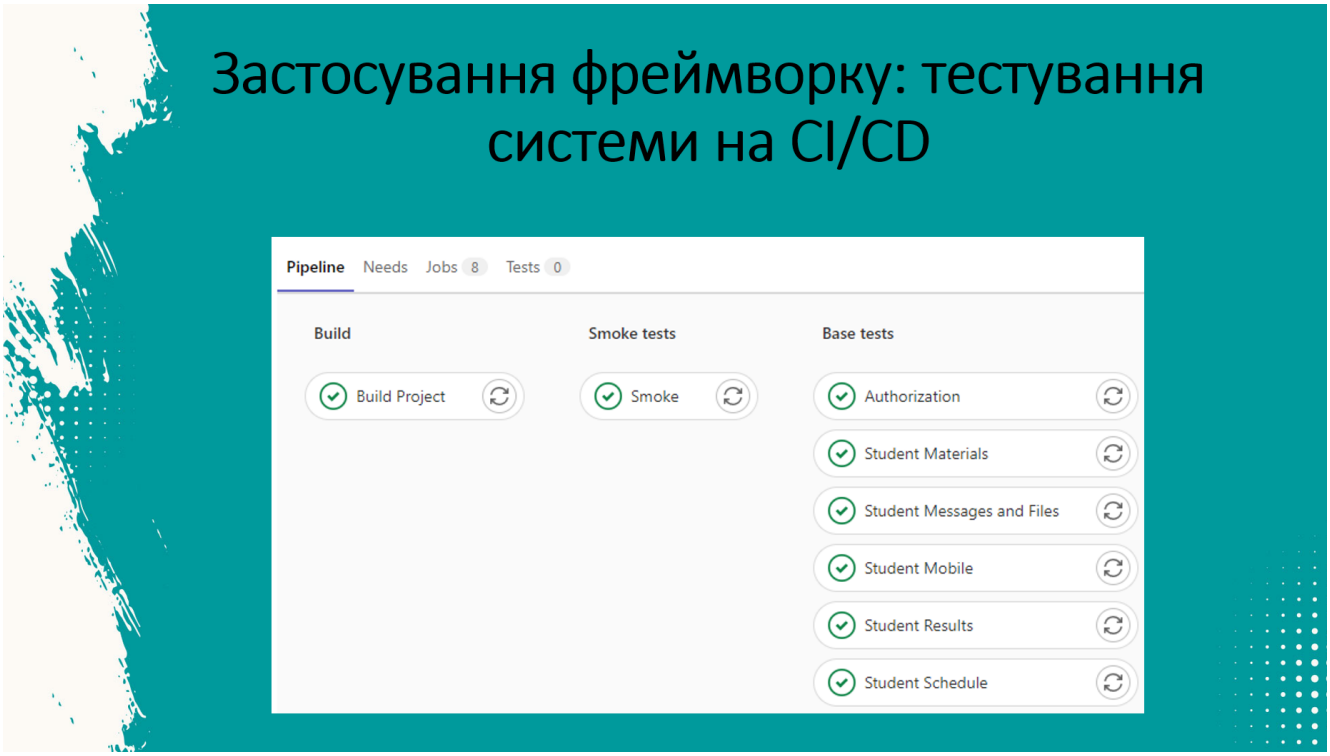


Рисунок Г.13 – Застосування фреймворку: тестування системи на CI/CD

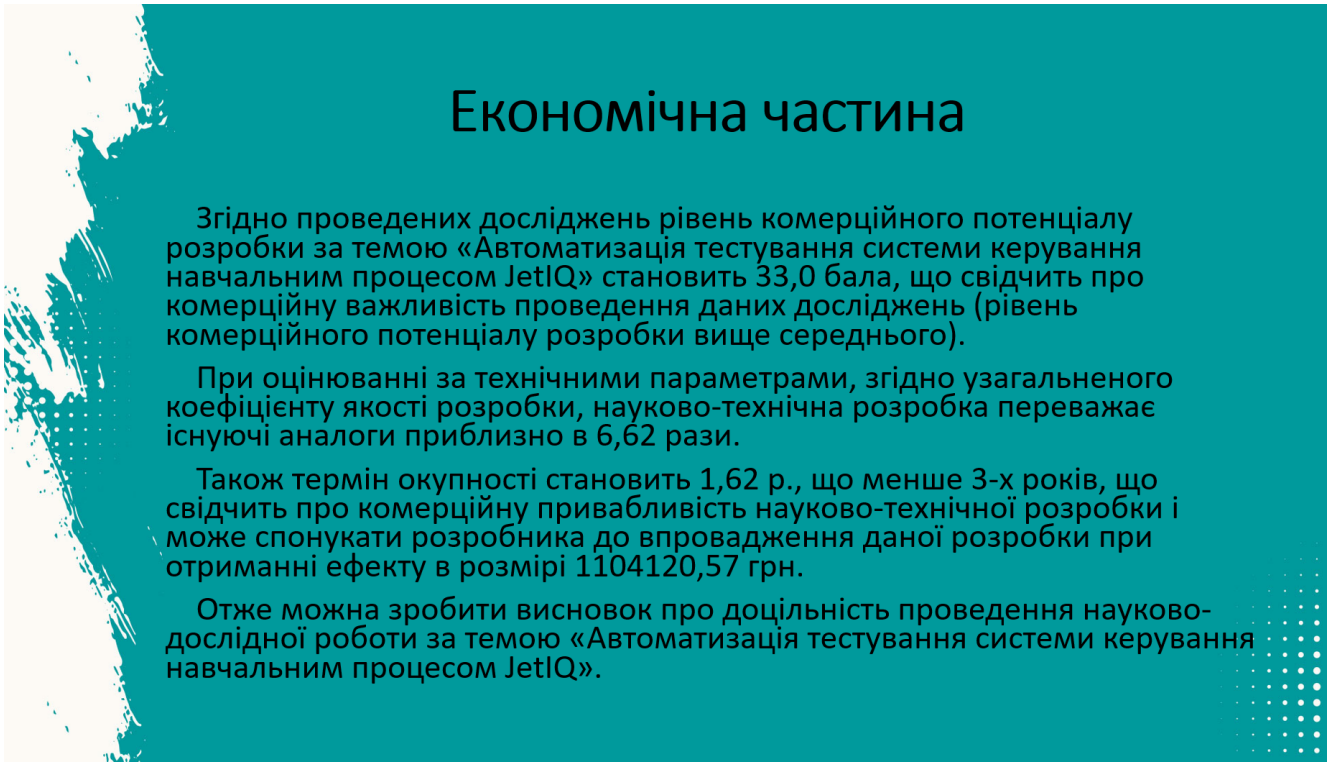


Рисунок Г.14 – Економічна частина

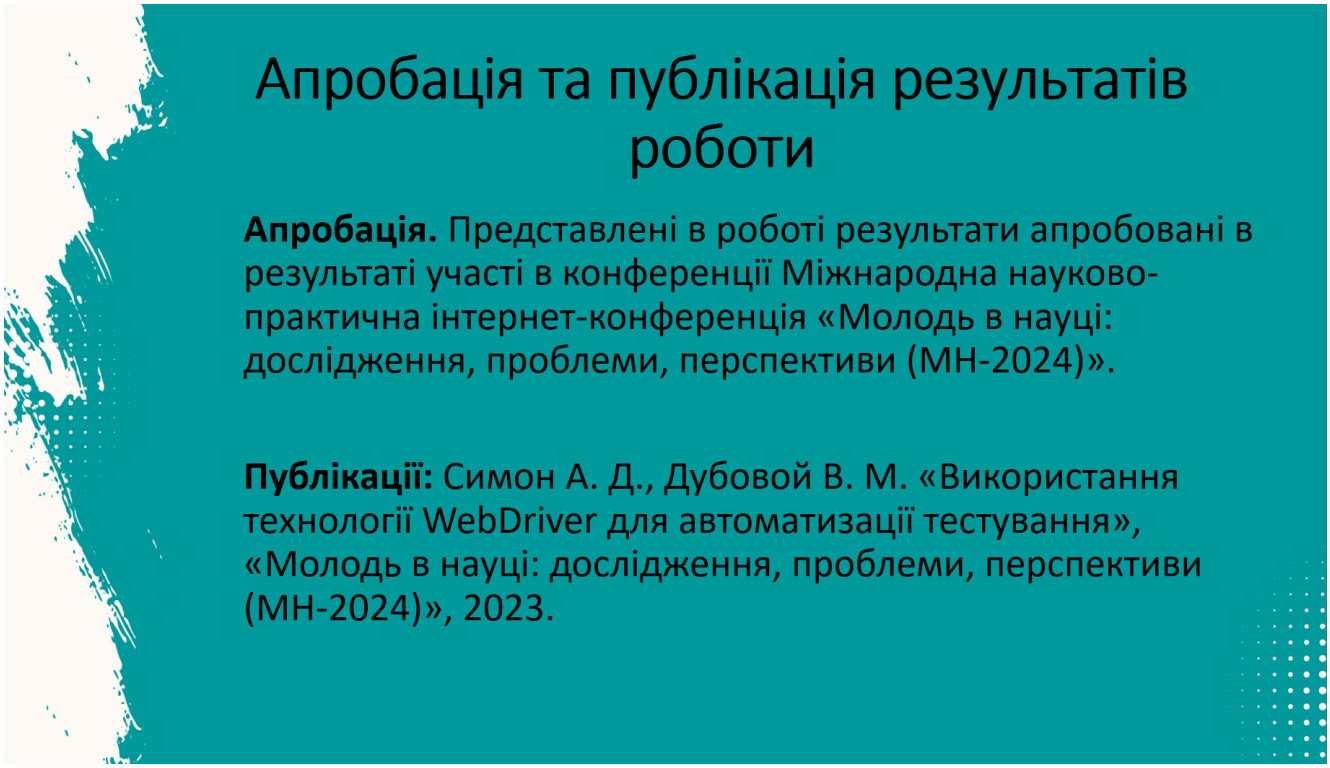


Рисунок Г.15 – Апробація та публікація результатів роботи

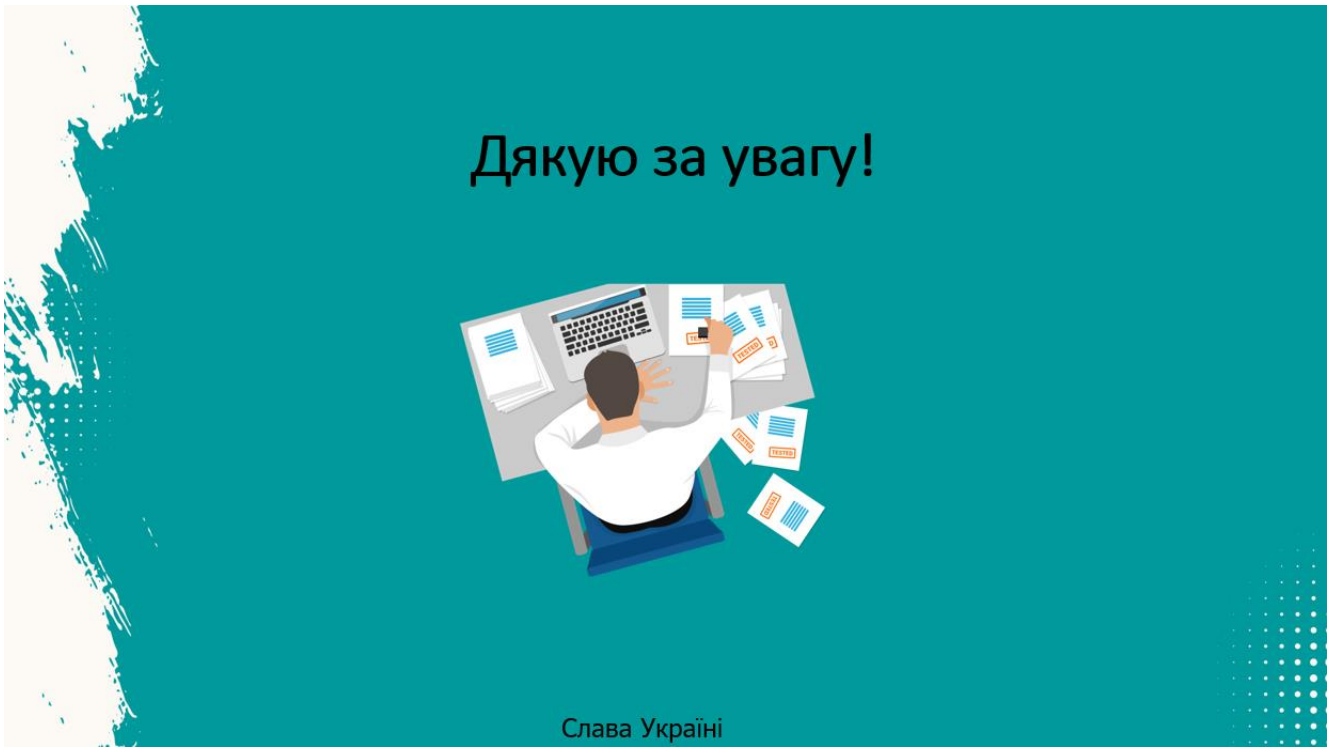


Рисунок Г.16 – Фінальний слайд