НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і

спеціалізованих комп'ютерних систем

«На правах рукопису»
УДК 004.415.538'24(043.3)

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системне програмування та спеціалізовані комп'ютерні системим»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Способи підвищення ефективності засобів тестування програмних систем»

Виконав:
студент ІІ курсу, групи КВ-21 мп
Варган Олексій Ігорович _____

Науковий керівник:
доцент кафедри СПСКС, к.т.н., доцент
Павловський Володимир Ілліч _____

Рецензент:
доцент кафедри ОТ, к.т.н., доцент
Волокита Артем Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп'ютерна інженерія»»

Освітньо-професійна програма «**Системне програмування та спеціалізовані комп'ютерні системи»**

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2022 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Варган Олексій Ігорович

1. Тема дисертації: «Способи підвищення ефективності засобів тестування програмних систем», науковий керівник дисертації Павловський Володимир Ілліч, доцент кафедри СПСКС, к.т.н., доцент, затверджені наказом по університету від «9» листопада 2023 р. №5217-с.

2. Термін подання студентом дисертації: 10 січня 2024 р.

3. Об'єкт дослідження: автоматизоване тестування веб-додатків.

4. Вихідні дані: фреймворк для автоматизованого тестування веб-додатків

5. Перелік завдань, які потрібно розробити: аналіз сучасних методів та інструментів автоматизованого тестування, визначення ключових вимог до фреймворку, розробка концептуальної та технічної моделі фреймворку, проведення випробувань і оцінка ефективності розробленого фреймворку.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: презентація.

7. Орієнтовний перелік публікацій: За тематикою проведених досліджень опубліковано 2 тезиси доповідей.

1.СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ V всеукраїнська науково-практична інтернет-конференція молодих вчених та студентів (30 листопада 2023 року), *Херсон, Україна*. [1]

2. ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ XVI науково-практична конференція магістрантів та аспірантів ПМК-2023 факультету прикладної математики (28 – 30 листопада 2023 року) *Київ, Україна*. [2]

9. Дата видачі завдання: 1 листопада 2022р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
	Постановка задачі та мети дисертації	01.11.2022р.	
	Огляд літературних джерел за тематикою дисертації	10.03.2023р.	
	Аналіз проблемної області	10.05.2023р.	
	Дослідження архітектури	10.06.2023р.	
	Розробка фреймворку автоматизованого тестування	20.11.2023р.	
	Розробка системи автоматизованого тестування	25.12.2023р.	
	Оформлення дисертації	07.01.2024р. – 10.01.2024р.	
	Підготовка презентації	08.01.2024р. – 10.01.2024р.	
	Захист магістерської дисертації	15.01.2024р. – 18.01.2024р.	

Студент Олексій ВАРГАН

Науковий керівник Володимир ПАВЛОВСЬКИЙ

РЕФЕРАТ

Актуальність теми.

Ця тема є актуальною через зростаючу залежність сучасних бізнес-структур та організацій від веб-додатків. В контексті постійного розвитку програмного забезпечення та веб-технологій, надійність і ефективність веб-додатків стають вирішальними. Критичний аналіз існуючих рішень у сфері автоматизованого тестування вказує на певні недоліки та обмеження, що вимагають інноваційного підходу. Відтак, дослідження направлене на розробку вдосконаленого фреймворку для автоматизованого тестування веб-додатків є важливим для подальшого розвитку IT-сектора та пов'язаних з ним галузей.

Мета і задачі дослідження

Мета: Розробка інноваційного фреймворку для автоматизованого тестування веб-додатків, який вирішує існуючі проблеми ефективності та масштабування.

Завдання:

Аналіз сучасних методів та інструментів автоматизованого тестування.

Визначення ключових вимог до фреймворку.

Розробка концептуальної та технічної моделі фреймворку.

Проведення випробувань і оцінка ефективності розробленого фреймворку.

Об’єкт дослідження: Процес автоматизованого тестування веб-додатків.

Предметом дослідження є фреймворк для автоматизації тестування, його структура та функціональність.

Наукова новизна полягає у способі підвищення ефективності тестування шляхом інтеграції автоматизації з передовими методологіями розробки та тестування програмного забезпечення CI/CD (Continuous Integration/Continuous Delivery) та BDD (Behavior Driven Development) що дозволяє зберігти багатофункціональність та мультиплатформість й при цьому вирішує проблему поширення подібних інструментів серед фахівців, що не володіють знаннями мов(и) програмування.

Практична цінність: Фреймворк має значний потенціал для використання у різних сферах, де потрібне надійне тестування веб-додатків, що сприяє підвищенню їх якості та надійності. Результати дослідження можуть бути впроваджені у практику ІТ-компаній.

Апробація роботи:

Дві теми одного із розділів магістерської дисертації будуть розміщені відповідно на двох наукових конференціях:

2.СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ V всеукраїнська науково-практична інтернет-конференція молодих вчених та студентів (30 листопада 2023 року), *Херсон, Україна*. [1]

3.ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ XVI науково-практична конференція магістрантів та аспірантів ПМК-2023 факультету прикладної математики (28 – 30 листопада 2023 року) *Київ, Україна*. [2]

Публікації.

За тематикою проведених досліджень опубліковано 2 наукові праці, а саме тези доповідей на 2-х конференціях.

Структура та обсяг дисертації.

Магістерська дисертація складається з 4 розділів, висновків, списку використаних літературних джерел (12 найменувань). Повний обсяг дисертації становить 97 сторінок, включаючи 90 сторінок основного тексту, 22 рисунків, 1 таблиці.

У *вступі* обґрунтовано актуальність проблеми, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів і їхнє впровадження.

У *першому* розділі розглянута загальна характеристика проблеми, проаналізовані існуючі розробки автоматизованого тестування веб-додатків, виділення переваг та недоліків розробленого фреймворку.

У *другому* розділі обґрунтовується вибір інструментів для розробки, описується логічна модель та архітектура фреймворку.

У *третьому* розділі формуються вимоги до системи і описується розробка фреймворку.

У *четвертому* розділі відбувається тестування додатку та аналіз отриманих даних.

У *висновках* виділяються результати дослідження, що сформовані на основі мети і завдання, що описані у вступі.

Ключові слова: автоматизоване тестування, веб-додатки, фреймворк, наукова новизна, практичне значення.

ABSTRACT

Relevance of the topic.

This topic is relevant due to the growing dependence of modern business structures and organizations on web applications. In the context of continuous development of software and web technologies, the reliability and efficiency of web applications become crucial. A critical analysis of existing solutions in the field of automated testing indicates certain shortcomings and limitations that require an innovative approach. Therefore, the research is aimed at developing an enhanced framework for automated testing of web applications, which is important for the further development of the IT sector and related industries..

Objective and Research Tasks.

Objective: Development of an innovative framework for automated testing of web applications that addresses existing efficiency and scalability issues.

Tasks:

Analysis of modern methods and tools for automated testing.

Identification of key requirements for the framework.

Development of conceptual and technical models of the framework.

Conducting tests and evaluating the performance of the developed framework.

Object of Research: The process of automated testing of web applications.

Subject of Research: The framework for testing automation, its structure, and functionality.

Scientific Novelty lies in the method of enhancing testing efficiency through the integration of automation with advanced software development and testing methodologies such as CI/CD (Continuous Integration/Continuous Delivery) and BDD (Behavior Driven Development). This approach enables the preservation of multifunctionality and cross-platform compatibility while addressing the challenge of spreading such tools among professionals who may not possess programming language expertise.

Practical Value: The framework has significant potential for use in various areas where reliable testing of web applications is needed, contributing to the improvement of their quality and reliability. The research results can be implemented in the practice of IT companies.

Research Validation:

Two topics from one of the sections of the master's dissertation will be presented at two scientific conferences:

1. "Modern Information Systems and Technologies V" - All-Ukrainian Scientific-Practical Internet Conference of Young Scientists and Students (November 30, 2023), *Kherson, Ukraine*. [1]
2. "Applied Mathematics and Computing XVI" - Scientific-Practical Conference of Master's and Ph.D. Students of the Faculty of Applied Mathematics (November 28-30, 2023), *Kyiv, Ukraine*. [2]

Publications.

Two scientific papers related to the research topic have been published, specifically abstracts of presentations at two conferences.

Structure and Volume of the Dissertation.

The master's dissertation comprises 4 chapters, conclusions, and a list of references (12 titles). The total volume of the dissertation is 97 pages, including 90 pages of the main text, 22 figures, and 1 table.

The introduction justifies the relevance of the problem, formulates the purpose and research tasks, demonstrates the scientific novelty of the obtained results, and outlines the practical significance of the work. Information about the validation of results and their implementation is also provided.

The first chapter examines the general characteristics of the problem, analyzes existing developments in automated testing of web applications, and highlights the advantages and disadvantages of the developed framework.

The second chapter justifies the choice of tools for development and describes the logical model and architecture of the framework.

The third chapter formulates system requirements and describes the development of the framework.

The fourth chapter covers the testing of the application and the analysis of the obtained data.

The conclusion section highlights the results of the research based on the formulated purpose and tasks outlined in the introduction..

Keywords: automated testing, web applications, framework, scientific novelty, practical significanc

ЗМІСТ

ВСТУП

Актуальність теми. Ця тема є актуальною через зростаючу залежність сучасних бізнес-структур та організацій від веб-додатків. В контексті постійного розвитку програмного забезпечення та веб-технологій, надійність і ефективність веб-додатків стають вирішальними. Критичний аналіз існуючих рішень у сфері автоматизованого тестування вказує на певні недоліки та обмеження, що вимагають інноваційного підходу. Відтак, дослідження направлене на розробку вдосконаленого фреймворку для автоматизованого тестування веб-додатків є важливим для подальшого розвитку IT-сектора та пов'язаних з ним галузей.

Мета і задачі дослідження.

Мета: Розробка інноваційного фреймворку для автоматизованого тестування веб-додатків, який вирішує існуючі проблеми ефективності та масштабування.

Завдання:

Аналіз сучасних методів та інструментів автоматизованого тестування.

Визначення ключових вимог до фреймворку.

Розробка концептуальної та технічної моделі фреймворку.

Проведення випробувань і оцінка ефективності розробленого фреймворку.

Об'єкт дослідження: Процес автоматизованого тестування веб-додатків.

Предмет дослідження: Фреймворк для автоматизації тестування, його структура та функціональність.

Методи дослідження. Використовуються методи аналізу та синтезу, моделювання, експериментальні методи (в тому числі тестування та оцінка розробленого фреймворку), а також методи системного аналізу для вивчення існуючих підходів і визначення оптимальних шляхів їх удосконалення.

Наукова новизна одержаних результатів. Розробка представляє собою оригінальний фреймворк, що інтегрує передові методи тестування і вирішує специфічні проблеми, які не були повністю розглянуті у попередніх розробках.

Практичне значення одержаних результатів. Фреймворк має значний потенціал для використання у різних сферах, де потрібне надійне тестування веб-додатків, що сприяє підвищенню їх якості та надійності. Результати дослідження можуть бути впроваджені у практику ІТ-компаній.

Особистий внесок магістранта. Специфічний внесок включає розробку концептуальної моделі фреймворку, його програмну реалізацію та проведення випробувань.

Апробація результатів дисертації. Дві теми одного із розділів магістерської дисертації будуть розміщені відповідно на двох наукових конференціях:

4.СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ V всеукраїнська науково-практична інтернет-конференція молодих вчених та студентів (30 листопада 2023 року), *Херсон, Україна*. [1]

2. ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ XVI науково-практична конференція магістрантів та аспірантів ПМК-2023 факультету прикладної математики (28 – 30 листопада 2023 року) *Київ, Україна*. [2]

Структура та обсяг дисертації. Магістерська дисертація складається з 4 розділів, висновків, списку використаних літературних джерел (12 найменувань). Повний обсяг дисертації становить 89 сторінок, включаючи 76 сторінок основного тексту, 15 рисунків, 1 таблиці.

Ключові слова: автоматизоване тестування, веб-додатки, фреймворк, наукова новизна, практичне значення.

1. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Техніко-економічна характеристика автоматизованого тестування веб-додатків

Вся інформація в Інтернеті зберігається на комп'ютерах, які називаються веб-серверами. На цих веб-серверах встановлено спеціальне програмне забезпечення, яке дозволяє користувачам шукати певну інформацію. Більшість цієї інформації представлена користувачам у формі веб-сайтів, кожен із яких має власне доменне ім'я чи адресу в Інтернеті. Для перегляду веб-додатків з будь-якого пристрою користувачі повинні встановити спеціальну програму - веб-браузер. Залежно від доменного імені веб-сайту, введеного в рядок пошуку, відповідна інформація відображається в області завдань браузера.

Усі веб-додатки складаються з однієї сторінки або кількох взаємопов'язаних веб-сторінок. Кожна з цих сторінок веб-програми складається з текстового файлу з розширенням *.html. Цей файл містить спеціальні текстові команди, так званий HTML-код, які визначають тип інформації, що відображається в браузері. Різноманітні графічні, аудіо- та відеодані не є частиною веб-сайту, а є окремими файлами, пов'язаними кодом HTML [1]. Бази даних використовуються для відображення інформації в будь-якій предметній області.

Предметна область — це частина реального світу, яка розглядається в рамках домену проектування та представлена у вигляді бази даних. Щоб описати певну область інтересів у формі бази даних, потрібно вирішити, яка інформація буде зберігатися та оброблятися в базі даних.

Усі веб-додатки є веб-сайтами, але не всі веб-сайти можна назвати веб-додатками. Основна відмінність між веб-сайтом і веб-додатком полягає в тому, що перший є інформаційним, тоді як другий є інтерактивним. Крім того, усі веб-сайти знову класифікуються як статичні та динамічні.

Статичний веб-сайт — це серія статичних HTML-сторінок, з'єднаних посиланнями.

Такі HTML-сторінки створюються вручну, завантажуються, зберігаються на віртуальному сервері, а потім відображаються в незмінному форматі кожного разу, коли користувач відвідує веб-сайт. Якщо потрібно змінити інформацію на статичному сайті, то потребується ручне редагування вихідного HTML-коду сторінки.

Виділяють кілька основних переваг та недоліків статичних сайтів. До переваг відносять:

- Навантаження на сервер мінімізовано;
- Швидке завантаження;
- дешева та швидка розробка;
- Легке перенесення з одного віртуального сервера на інший.

Основним недоліком таких сайтів є те, що їх важко оновлювати та вносити зміни. Пересічний користувач не може керувати таким сайтом без певних знань веб-програмування.

Динамічні веб-сайти мають багато різних функцій порівняно з статичними.

Наприклад, можна редагувати інформацію або публікувати щоденні новини чи блоги без будь-яких знань веб-програмування, але якщо основною діяльністю користувача на цьому веб-ресурсі є перегляд інформації, то цей ресурс вважається веб-сайтом, а не мережею.

Динамічні веб-сайти набагато гнучкіші у використанні, ніж статичні веб-сайти. Такий веб-сайт можна легко розробити з нуля, написавши вручну весь необхідний програмний код, скрипти тощо. Для розробки динамічних веб-сайтів найчастіше використовуються спеціальні системи керування контентом (CMS). Такі системи дозволяють використовувати готові модулі та скрипти. На основі такої CMS можна створювати будь-яку кількість динамічних сайтів. Найпопулярнішими CMS сьогодні є Wordpress, Joomla та Bitrix. Основною характеристикою веб-програм є те, що вони інтерактивні. Це означає, що користувачі є активними учасниками, а не пасивними споживачами. Користувачі постійно шукають інформацію, натискають клавіші, взаємодіють з клавіатурою та

мишею.

Прикладом веб-програми є сайт instagram.com.

У ньому користувачі не тільки переглядають інформацію, а й постійно виконують різноманітні дії, наприклад, ставлять лайк , записують відео, публікують нове фото тощо.

«Мета процесу кваліфікаційного тестування системи — перевірити виконання кожної системної вимоги та переконатися, що система готова до доставки». [4].

«Мета процесу підтримки прийняття програмного забезпечення полягає в тому, щоб переконати покупця, що продукт відповідає визначеним вимогам».[4]

Таким чином, обидва етапи – і кваліфікаційне тестування та приймання служать подібним цілям - перевірка готовності ПЗ та відповідності його вимогам. Відповідно до стандарту IEEE SWEBOK v3.0 «тестування програмного забезпечення (Software Testing) – це динамічна перевірка відповідності між реальною та очікуваною поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином» [12].

Тестування є однією зі складових процесу контролю якості і включає в себе:

- планування процесу тестування (Test Management);
- розробку та проектування тестів (Test Design);
- прогін тестових сценаріїв (Test Execution);
- аналіз результатів, одержаних після прогонів тестів (Test Analysis).

Відповідно до посібника зі стандартизації та сертифікації програмного забезпечення «всі види тестування програмного забезпечення, залежно від переслідуваних цілей, можна умовно розділити на такі групи:

- функціональні;

- дисфункційні;

- пов'язані із змінами» [7].

Функціональне тестування базується на функціональних можливостях, функціях і взаємодії з іншими системами, включаючи компоненти або модулі

(Component/Unit testing), інтеграцію (Integration testing), системи (System testing) і приймальний рівень (Acceptance testing).

Функціональні види тестування розглядають зовнішню поведінку системи. Нижче наведено найпоширеніші види функціональних тестів:

- тестування безпеки (Security and Access Control Testing);

- тестування взаємодії (Interoperability Testing) [7].

Нефункціональні тести відносяться до тестів, які необхідні для визначення характеристик програмного забезпечення та можуть бути виміряні в різних кількостях.

Нижче наведено основні види нефункціональних тестів:

–навантажувальне тестування (Performance and Load Testing);

–стресове тестування (Stress Testing);

–тестування стабільності або надійності (Stability/Reliability Testing);

–об'ємне тестування (Volume Testing);

–тестування зручності користування (Usability Testing);

–тестування на відмову та відновлення (Failover and Recovery Testing);

–конфігураційне тестування (Configuration Testing)»[7].

Автоматизоване тестування програмного забезпечення – це процес перевірки програмного забезпечення, в якому основні функції та кроки тесту, такі як запуск, ініціалізація, виконання, аналіз і вихід, виконуються автоматично за допомогою інструменту автоматизованого тестування.

Спеціаліст з автоматизованого тестування програмного забезпечення (Software Automation Tester) — технічний фахівець (тестер або розробник програмного забезпечення), який забезпечує створення, налагодження та підтримку робочих умов тестових скриптів, наборів тестів і інструментів для автоматизованого тестування.

Інструмент для автоматизованого тестування (Automation Test Tool) - це програмне забезпечення, яке використовується спеціалістами з автоматизованого тестування для створення, налагодження, запуску сценаріїв тестування та аналізу

результатів виконання.

Тестовий Скрипт (Test Script) - це набір інструкцій для автоматичної перевірки певної частини програмного забезпечення.

Тестовий Набір (Test Suite) - це комбінація тестових скриптів для перевірки певної частини програмного забезпечення, об'єднаної загальною функціональністю або цілями, що переслідуються запуском даного набору.

Тести для запуску (Test Run) - це комбінація тестових скриптів або наборів тестів для подальшого спільного запуску (послідовного або паралельного, залежно від призначення та функціональних можливостей засобу автоматизованого тестування) для виконання перевірки на певній версії продукту [8].

Для більш ефективного написання та роботи вище зазначених артефактів доцільно використовувати якийсь загальний фреймворк автоматизованого тестування.

Однак в такому випадку, необхідно також буде вирішити одну з найбільших проблем в автоматизації тестування програмного забезпечення - підготовка та відстеження вже зазначених артефактів, а також налаштування середовища, необхідного для виконання автоматизованих тестів. З іншого боку, може бути брак кваліфікації та досвіду щодо умов і середовищ розробки, що ускладнює залучення якомога більшої кількості спеціалістів із тестування до процесу підтримки автоматизованого тестування.

Більша частина автоматизованих тестів вимагають розуміння якоїсь з скриптовох мов для їх написання, таких як VB Script, Java Script і т.д.. Як правило, цей інструмент дозволяє створювати тести за допомогою запису та відтворення. Однак такі сценарії зазвичай не дуже ефективні, їх не можна використовувати багаторазово, і їх важко підтримувати. Платформа автоматизованого тестування — це набір правил, концепцій і методів, спрямованих на покращення повторного використання, зниження витрат на обслуговування та підвищення надійності використання тестів. Використання фреймворку вирішує більшість описаних проблем. Це означає, що інструмент використовується широким колом

професіоналів, включаючи розробників і ручних тестувальників.

1.2 Аналіз існуючих розробок автоматизованого тестування веб-додатків

У цьому розділі наведено порівняння розробленого фреймворку з існуючими на ринку поширеними рішеннями в даній галузі. Як критерії для аналізу ринку існуючого ПЗ для автоматизації тестування використовуватимуться:

1. Поширеність інструменту – це означає існування бази користувачів, де можна знайти відповіді на запитання, що виникають у результаті використання інструменту. Знання також більш імовірно будуть актуальними для розробників. Важливість цього критерію полягає в тому, що розробники Autotest мають можливість скористатися накопиченим досвідом інших розробників і не повинні вирішувати кожну проблему з нуля.

2. Підтримка – тобто, як часто виходять оновлення/виправлення помилок. Цей критерій важливий для можливості виправлення помилок, що впливають на стабільність роботи системи.

3. Ціна інструменту – варіанти платного програмного забезпечення, безкоштовного програмного забезпечення та умовно безкоштовного програмного забезпечення. Ціна засобу є економічним критерієм, який визначає безпосередній розрахунок ефективності використання програмних засобів автоматизації.

4. Відкритість вихідного коду – відкритий вихідний код або власний формат програмного забезпечення. За цим критерієм можна оцінити можливість адаптації інструменту до індивідуальних потреб.

5. Підтримка браузерів – оскільки ця структура спеціально спрямована на тестування веб-додатків, підтримка нових версій популярних браузерів відіграє важливу роль.

6. Підтримка операційних систем – якщо програмне забезпечення, що тестується, є мультиплатформним, підтримка якомога більшої кількості операційних систем впливатиме на вибір інструменту.

7. Мова програмування – мова, якою написана структура, впливає на легкість/складність пошуку експертів для розробки та підтримки тестів компанії та самої основи.

8. Можливості взаємодії зі сторонніми бібліотеками у рамках Continious Integration/Delivery – цей критерій залежить від того, чи можна інтегрувати тестування в безперервний процес розробки та доставки програмного забезпечення.

9. Можливість роботи з моделлю BDD (Behavior Driven Development) - цей стандарт відповідає за вирішення проблеми відсутності необхідного досвіду розробки програмного забезпечення серед користувачів фреймворку.

10. Простота установки, налаштування та зручність користування. Аналогічно пункту 9 цього списку, простота використання впливатиме на розширення списку користувачів фреймворку.

Таблиця 1.1 - Порівняння фреймворків автоматизації тестування

Категорія/Продукт	Розроблений фреймворк	Selenium	Katalon	UFT	Test Complete	Watir
Поширеність	-	+++	++	+	++	++
Підтримка Інструменту	+++	+++	+	+	++	+
Ціна	Безкоштовний /внутрішні витрати	Безкоштовний	Безкоштовний	Дорогий	Середня ціна	Безкоштовний
Відкритість	+++	+++	-	-	-	+++
Підтримка браузерів	IE, Chrome, Firefox, Opera, Edge, мобільні браузери	IE, Chrome, Firefox, Opera, Edge, мобільні браузери	IE, Chrome, Firefox	IE, Chrome, Firefox, Safari, мобільні браузери	IE, Chrome, Firefox, Opera, Safari, Edge	IE, Chrome, Firefox, Safari, Edge
Підтримка ОС	Windows, Linux OS X	Windows, Linux OS X	Windows, Linux OS X	Windows	Windows	Windows, Linux OS X

Мова Програмування	Java	Java, C#, Perl, Python, JavaScript, Ruby, PHP	Java/Groovy	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++, C#	Ruby
CI/CD	+++	+++	++	+++	+++	+
BDD	+++	-	-	-	++	+++
Простота та зручність	+++	-	++	-	+	-

Умовні позначення:

+/++/+++ - відповідність категорії/критерію. Чим більше плюсів, тим краще.

- невідповідність критерію/відсутність функціональності.

Основними перевагами розробленого фреймворку в порівнянні з наявними на ринку продуктами, будуть:

Підтримка інструменту.

Підтримка співробітників компанії забезпечує швидке вирішення проблем і покращення.

Умовна безкоштовність.

Витрати включають лише початкові витрати на розробку та обслуговування та витрати на розробку нових функцій. Перевага такого підходу в тому, що для розвитку використовуються внутрішні ресурси. Це означає, що будуть реалізовані лише необхідні функції і не будуть запроваджені регулярні обов'язкові платежі.

Відкритість.

Проект планує реалізувати свої ініціативу як вільне програмне забезпечення з вихідними кодами. Це надає можливість для широкого розповсюдження розробленого інструменту. Завдяки цьому, проект може отримати підтримку від сторонніх розробників, які бажають зробити свій внесок у відкритий проект. Також це може призвести й до покращення репутації оригінального розробника, завдяки

участі в розвитку відкритих технологій.

Підтримка браузерів.

Фреймворк взаємодіє з практично всіма наявними на ринку браузерами, оскільки базується на Selenium Webdriver. Він підтримує як власні драйвери, так і ті, які розроблені сторонніми розробниками. Також є можливість створення власного драйвера для найспецифічніших і навіть пропрієтарних версій браузерів.

Мова програмування та підтримка ОС.

Для розробки фреймворку була обрана мова програмування Java, яка наразі є найбільш поширеною за низкою індексів. Наприклад, згідно із відомим індексом TIOBE Programming Community Index на травень 2018 року, мова Java посідає перше місце з рейтингом 16,38% [1]. Таким вибір дозволяє уникнути проблеми зі складнощами пошуку нових розробників для проекту. Крім того, це також впливає на потенційну кількість розробників і тестувальників, які оберуть цей фреймворк для своїх цілей.

CI/CD.

Фреймворк розробляється у формі бібліотеки, яку можна легко підключити до свого проекту шляхом вказання залежностей у настроювальних файлах систем Maven, Ant, Gradle. Це робить процес вбудовування в проект дуже простим і сприяє зручності інтеграції з середовищами CI/CD, такими як Jenkins, Hudson та ін.

BDD.

Фреймворк може здобути ще більше популярності завдяки можливості створювати автоматизовані тести за допомогою технології Behavior Driven Development (BDD). Це дозволяє фахівцям у галузі тестування програмного забезпечення, навіть тим, хто не має досвіду з жодною мовою програмування, аналізувати існуючі тести та створювати нові, використовуючи готові базові кроки.

При цьому необхідно також відзначити і недоліки рішення, що розробляється:

Поширеність.

На ранніх стадіях розробки проект відомий лише вузькому колу штатних

розробників і обмеженій кількості користувачів, які відкрили репозиторій проекту з відкритим кодом. Тому цей проект підтримується лише внутрішніми розробниками. Цей недолік буде виправлено, коли інструмент стане більш доступним.

Мова програмування.

Хоча Java наразі є найпоширенішою мовою, вона також може обмежити кількість користувачів, які віддають перевагу іншим популярним мовам для розробки автоматизованих тестів програмного забезпечення, таким як Python, Ruby або C#. Цей недолік частково усувається завдяки використанню інструментів технології BDD, які частково усувають потребу в спеціалізованому персоналі зі знанням мови Java.Однак повністю усунути такі потреби неможливо.

1.3 Постановка задачі розробки фреймворку автоматизованого тестування

З кожним роком вимоги інтернет-користувачів до Web-ресурсів зростають, звичайних статичних або динамічних сайтів стає мало, і на перший план виходять складні Web-додатки з великим функціоналом, які задовольняють безліч запитів споживача. Оскільки метою цієї дипломної роботи є створення саме Web-програми, тобто простим HTML, CSS і JS кодом не обійтися. Потрібен такий інструмент, який дозволить робити безліч операцій без перезавантаження сторінки та створити програмний код, який буде строго структурований і його буде легко підтримувати. На сьогоднішний день найпопулярнішими вважаються «Angular.js» (рис. 1.1) і «React.js» (рис. 1.2) [7, с. 275].

Рисунок 1.1. Фреймворк Angular.js

Рисунок 1.2. Фреймфорк React.js

Розглянемо, який із цих фреймворків найбільше задовольнятиме наші потреби.

Angular.js зараз вважається найпопулярнішим фреймворком, розробка якого безпосередньо підтримується компанією Google. React.js у свою чергу гарний з погляду продуктивності, бо в ньому є Virtual DOM. Ця властивість може оновлювати HTML-код сторінки, порівнюючи різницю між попередньою та поточною версією коду. Враховуючи те, що нам дана властивість не потрібна, виберемо фреймворк Angular.js, оскільки він поєднав у собі відмінний набір комбінацій низького порога входження і величезного набору функцій. Щоб почати використовувати Angular.js у своїй роботі, потрібно не більше години. На даний момент ще немає жодного завдання під час розробки, яку не вдалося б вирішити за допомогою даного фреймворку. Тож простота та надійність однозначно перемагає.

Ключові функції розроблюваного фреймворку автоматизації тестування:

1. Keyword Driven підхід

Якщо тестувальники не є експертами скриптових мов, слід мати можливість замінити скрипти тестами на основі ключових слів (keyword-driven). Цей підхід дозволяє тестувальнику (автоматизатору) створювати автоматичні тести, просто описуючи кожен крок тесту. Наприклад, при тестуванні процесу авторизації, де користувач повинен запустити програму, ввести своє ім'я користувача та пароль, а потім натиснути кнопку входу. При звичайному підході тестувальникам необхідно було писати скрипт обраною мовою програмування, наприклад Java, Python, VB Script, який би запустив програму, розпізнав кожен об'єкт на екрані (поля введення імені, пароля, кнопку входу), далі вводив ім'я користувача, пароль, та натискав

кнопку входу. При використанні підходу Keyword Driven, тестувальнику не обов'язково розуміти скриптову мову для того, щоб створити такий тест, йому достатньо описати ці події (запуск програми, введення імені користувача "abc", введення пароля "xxx", натискання кнопки "вхід") мовою, максимально наближеному до природного. Цей підхід автоматизації спрощує роботу фахівців із тестування. Ітерація наборів даних

Ця практика дозволяє тестувальникам використовувати одні й самі автоматичні тести, але запускати їх із різними наборами тестових даних. Наприклад, можна використовувати один тестовий сценарій для входу в систему, проте виконувати його з різними комбінаціями імені користувача та пароля, щоб протестувати різні сценарії вхідних даних. Якщо фреймворк виявляється досить гнучким і дозволяє визначати кілька ітерацій для запуску з різними наборами даних, це може суттєво зменшити час, який витрачається на автоматизацію.

2. Розподілений запуск на кількох комп'ютерах

Корисною функцією є можливість розподіленого запуску на кількох комп'ютерах. Наприклад, це корисно, якщо компанія має потребу у створенні лабораторії QA з кількома серверами.

3. Автоматичний запуск за часом

Для зручності контролю та планування запусків слід передбачити можливість запускати їх автоматично у призначений час. Фреймворк повинен мати можливість автоматичного запуску в заданий час (щодня/тижня), мати можливість повторів (наприклад, запускати щодня о 6 годині). Цей функціонал може бути реалізований також на базі системи Continuous Integration, наприклад Jenkins/Hudson.

4. Звіти результатів запуску

Під час розробки фреймворку необхідно продумати механізм зручних та зрозумілих користувачам звітів щодо прогонів тестових сценаріїв. У цьому випадку можлива потреба здійснення зв'язку звітності, яку видає фреймворк, з уже існуючим інструментом управління тестуванням для ручного тестування. При цьому слід зважити на можливість інтеграції результатів автоматичних тестів зі звітами

інструментів тест менеджменту. Таким чином, буде забезпечено можливість узагальнено аналізувати результати активностей з ручного та автоматичного тестування.

Висновки до розділу

У цьому розділі наведено порівняння розроблюваного фреймворку з існуючими на ринку поширеними рішеннями в даній галузі за описаними критеріями.

Показано: В розділі детально розглянуто структуру та особливості веб-додатків, включаючи їх зберігання на Web-серверах та взаємодію з користувачами через Web-браузери. Визначено основні компоненти Web-додатків, такі як HTML-сторінки та зовнішні файли (графіка, аудіо, відео), а також роль баз даних у відображенні інформації про предметну область. Описано різницю між статичними та динамічними веб-сайтами, включаючи їх переваги та недоліки.

Запропоновано: На підставі аналізу веб-додатків, виявлено потребу у розробці більш ефективних методів автоматизованого тестування, які б враховували специфіку цих додатків. Зокрема, важливість врахування динамічної природи веб-додатків та необхідності автоматизації процесів тестування для підвищення продуктивності розробки та забезпечення високої якості кінцевого продукту.

Ці висновки узагальнюють аналіз проблемної області та підкреслюють важливість розробки фреймворку для автоматизованого тестування веб-додатків, що буде розглянуто далі у роботі.

2. РОЗРОБКА ФРЕЙМВОРКУ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

2.1 Автоматизоване тестування

На підставі пропозицій, для збільшення ефективності діяльності відділу тестування розроблено модель «Як має бути» з використанням фреймворку автоматизації тестування. Її схема та декомпозиція процесу представлені на рисунках 2.1. та 2.2..

Рисунок 2.1. Схема процесу роботи відділу тестування «Як має бути»

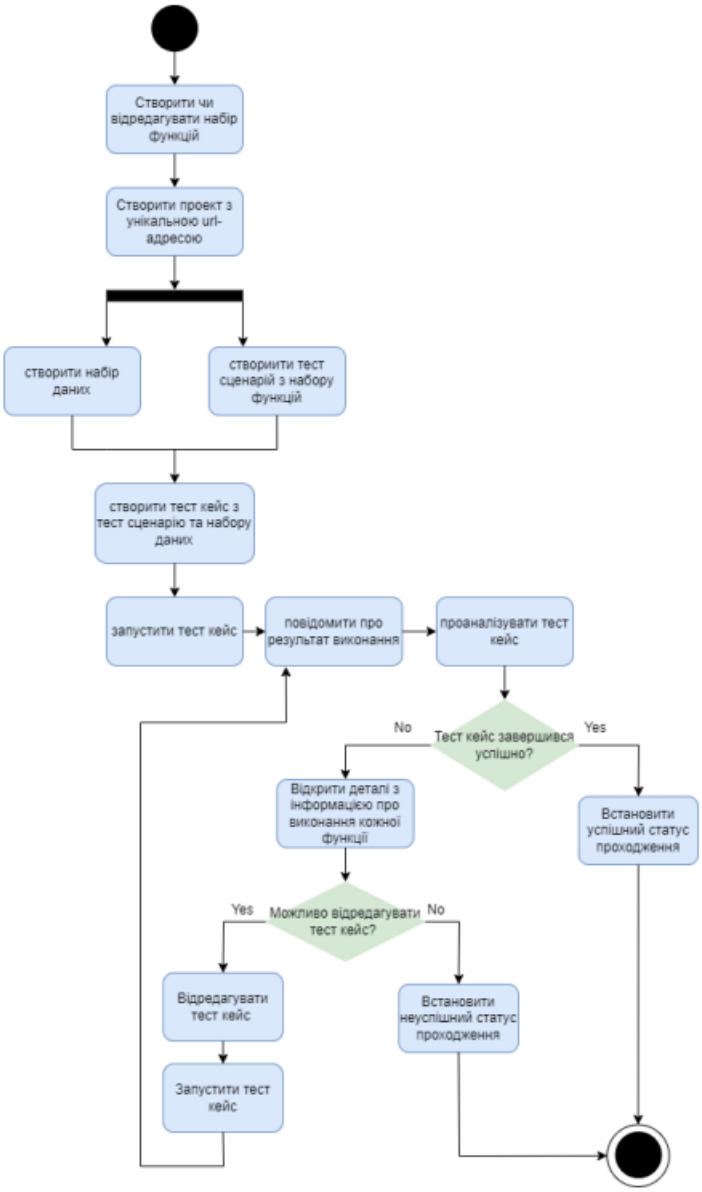


Рисунок 2.2. Декомпозиція схеми процесу роботи відділу тестування "Як має бути"

Фреймворк буде використаний для розробки та прогонів тестів на стадіях smoke/sanity тестування, регресійного тестування, а також за бажанням замовника (зовнішнього чи внутрішнього) на стадії приймального тестування/приймальних випробувань.

Це дозволить скоротити час, витрачений фахівцями з тестування на прогін тестів, які проводилися в ручному режимі, а також забезпечить можливість виключення розробників із участі у процесі на етапі smoke/sanity тестування, оскільки перевірка працездатності версії буде проводитися автоматично.

Також фреймворк допоможе у скороченні часу на підготовку звітів про прогони тестів, оскільки цей процес буде відбуватися автоматично, і звіти будуть вивантажуватись у зручній формі на спеціальний портал, з можливістю зв'язування з інструментами Continuous Integration/Continuous Deployment. Таким чином, після впровадження фреймворку автоматизації тестування, виявлені недоліки будуть усунені, а при необхідності напрацювання з впровадження автоматизованих тестів можуть бути передані замовнику для підвищення ефективності приймального тестування на апаратному оточенні замовника.

2.2 Концептуальна модель процесу тестування програмного забезпеченняс

В результаті аналізу діяльності з тестування програмного забезпечення була розроблена модель IDEF0 «Як є». Існуюча технологія тестування передбачає ручне тестування, як нового функціоналу, так і смоук і регресійних тестів, що потребує значних людських ресурсів. Схема IDEF0 діяльності відділу тестування ПЗ та декомпозиція поточних етапів/операцій діяльності відділу тестування DFD представлена на рисунках 2.3 та 2.4.

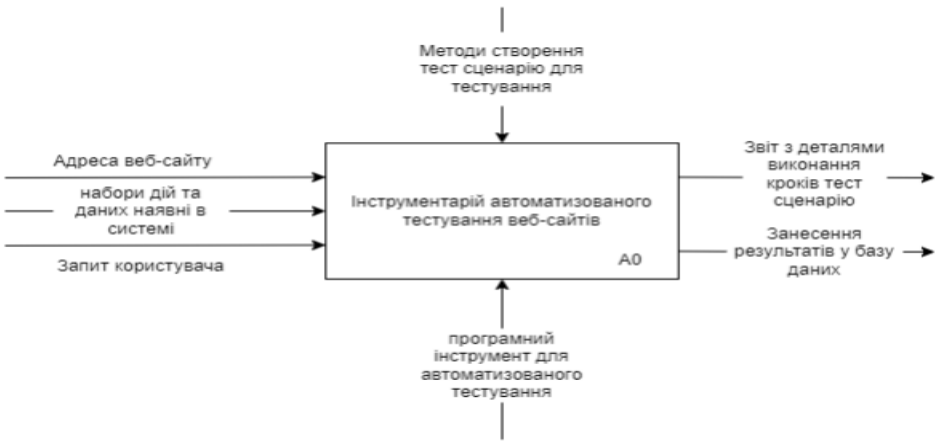


Рисунок 2.3. Схема процесу роботи відділу тестування «Як є»

Рисунок 2.4. Декомпозиція схеми процесу роботи відділу тестування «Як є»

Виявлені недоліки за даною схемою процесу:

– низька продуктивність праці фахівців із тестування;

– низька якість звітної інформації через можливі помилки фахівців при ручному відображенні результатів прогонів тестів;

– висока трудомісткість тестування великих релізів;

– недосконаліть організації первинних тестових даних різними співробітниками;

– недосконаліть процесу аналізу проблем і помилок, що виникають при тестуванні;

– розробникам доводиться брати участь у процесі тестування на етапі смоук тестування, оскільки встановлення ПЗ, що поставляється, проводиться в ручному режимі без використання методики Continuous Integration/Delivery.

Замовник представлений на схемі як учасник приймального тестування продукту. Проблема підвищення ефективності приймального тестування також можна розглядати у межах проблематики цієї роботи.

2.3. Розробка логічної моделі автоматизованого тестування

Для створення об'єктної моделі фреймворку було обрано мову UML (Unified Modeling Language).

Ця мова дозволяє у вигляді стандартизованих креслень та схем візуалізувати, специфувати, конструлювати та документувати програмні системи. Мова придатна для складання моделей систем будь-якого масштабу та напряму, її застосовують при моделюванні інформаційних системи великих та малих підприємств, Web-додатків, вбудованих систем реального часу.

Для побудови логічної моделі даних фреймворку було обрано методологію IDEF1x, яка застосовується для створення реляційних моделей даних.

Далі подано перехід від структурної діаграми «як має бути», описаної в розділі 1 до діаграми варіантів використання (Use case Diagram), яка також має назву – діаграма прецедентів.

Посібник користувача UML стверджує: "Суб'єкт (актор) - це будь-яка сутність, яка взаємодіє зовні із системою, або логічно пов'язаний набір дій, які виконуються під час взаємодії з прецедентом. До суб'єктів відносяться не тільки люди, а й технічні пристрої, програми та інші системи, які можуть створювати самі розробники та які можуть бути джерелом впливу на систему [2].

Прецеденти (use case) – це опис безлічі послідовностей дій, включаючи їх варіанти, які виконуються системою з метою отримання актором певного результату, важливого для нього. Варто зауважити, що в цьому описі не розглядається конкретний механізм взаємодії суб’єктів із системою, що становить одну з ключових особливостей розробки прецедентів. Типовим представленням прецеденту на діаграмах є еліпс, що містить коротку назву відмінка або назву форми дієслова з описовим словом.

Діаграма використання прецедентів призначена для опису функціонального призначення системи. Діаграма прецедентів фреймворку автоматизації тестування наведено рисунку 2.5.

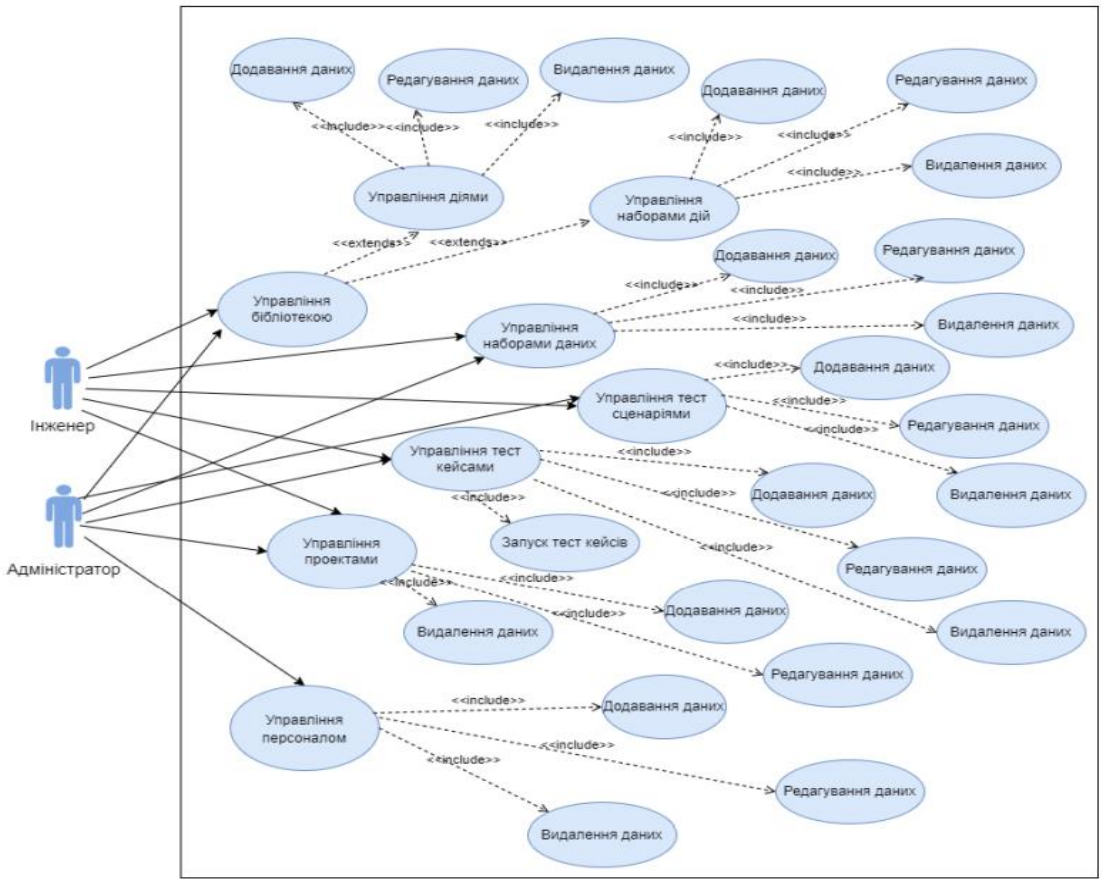


Рисунок 2.5. Діаграма прецедентів використання фреймворку автоматизації

Наступним кроком буде розробка діаграми класів для фреймворку (Class Diagram). У термінах інформаційних систем, клас виступає основним елементом.

Це поняття є ключовим в об'єктно- орієнтованих мовах програмування. Важливо відзначити, що існує відповідність між класами на UML-діаграмах і класами у програмному коді. Більшість інструментів для створення UML-діаграм дозволяють генерувати програмний код для обраної мови програмування на основі діаграми класів.

Кожен клас має обов'язкову унікальну назву, а також включає в себе атрибути та методи. Атрибути представляють властивості класу, в той час як методи описують можливі дії, які об’єкт даного класу може виконувати. Зазвичай методи призначені для зміни властивостей класу.

Абстрактна діаграма класів для фреймворку, що розробляється, представлена на рисунку 2.6.

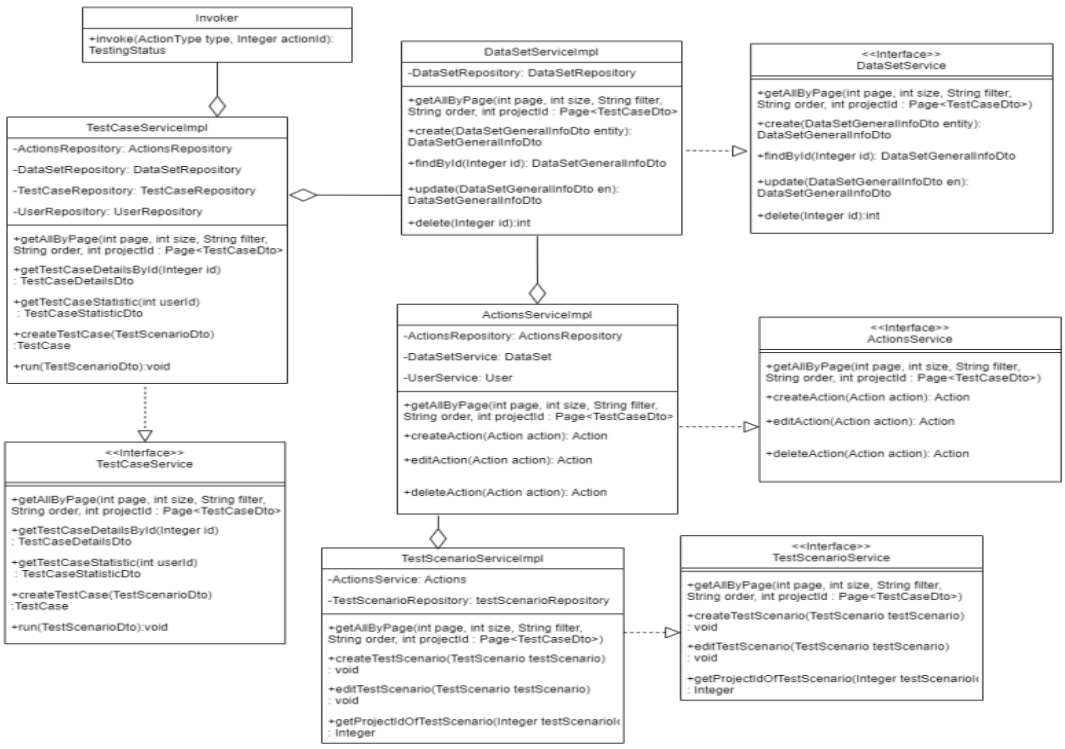


Рисунок 2.6. Анотація діаграма класів фреймворку автотестування

З цієї діаграми видно, які процеси будуть представлені в фреймворку, що розробляється, і їх зв'язки.

2.4 Архітектура фреймворку автоматизованого тестування

У сучасному світі програмне забезпечення використовується практично у всіх сферах життя, гігантські суми витрачаються на розробку різноманітних програм, потрібних у промисловості, бізнесі, індустрії розваг, освіті та медицині [1]. Завдання зниження вартості розробки програмного забезпечення та поліпшення якості продукції є однією з найбільш актуальних в індустрії інформаційних технологій.

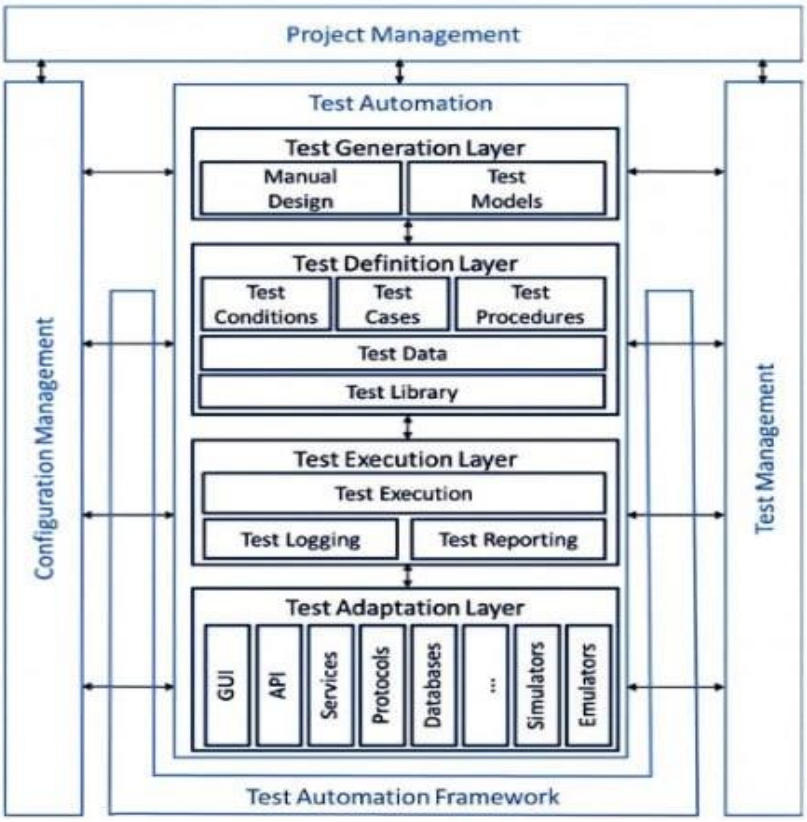


Рисунок 2.7. Структура тестового фреймворка

Розглянемо прикладні модулі, які можуть входити до складу фреймворку для автоматизованого тестування веб-додатків:

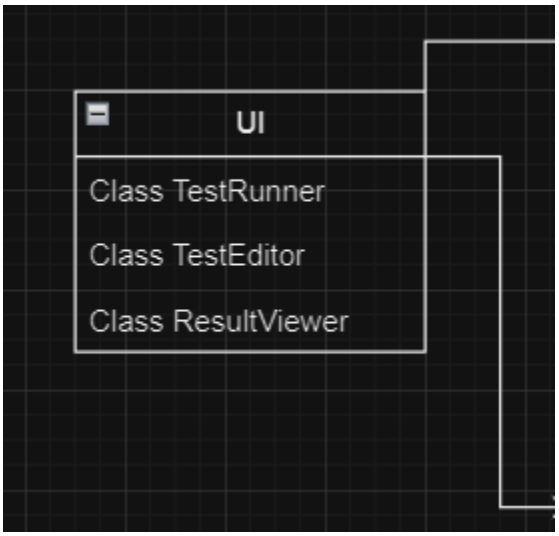


Рисунок 2.8 Модуль інтерфейсу користувача

Модуль інтерфейсу користувача (UI): Цей модуль забезпечує взаємодію між користувачем та фреймворком. Він може включати інструменти для створення, редагування та запуску тестових сценаріїв, а також відображення результатів тестування. Цей модуль забезпечує інтуїтивно зрозумілий інтерфейс,

що дозволяє користувачам легко взаємодіяти з функціями фреймворку.

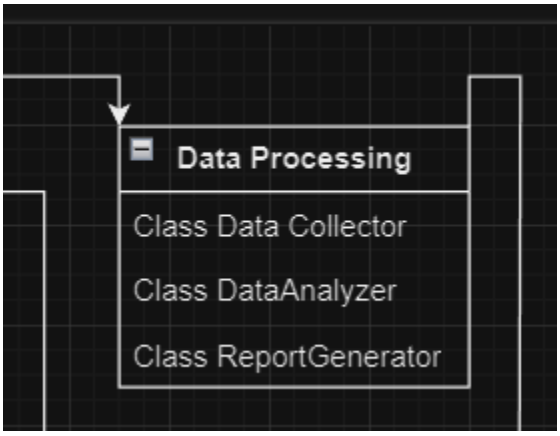


Рисунок 2.9 Модуль обробки даних

Модуль обробки даних: Відповідає за збір, обробку та аналіз даних, отриманих в результаті тестування. Цей модуль може включати алгоритми для обробки логів, збір статистики, а також забезпечення агрегації та візуалізації даних тестування.

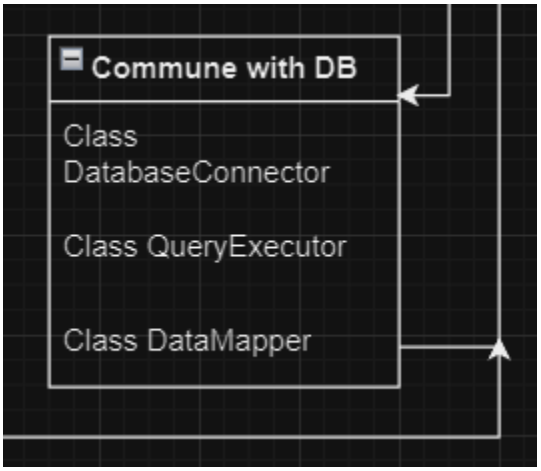


Рисунок 2.10 Модуль комунікації з базами даних

Модуль комунікації з базами даних: Цей модуль забезпечує взаємодію фреймворку з зовнішніми базами даних, які можуть зберігати тестові сценарії, результати тестування або іншу важливу інформацію. Він може включати функції для здійснення запитів до баз даних, отримання та збереження даних.

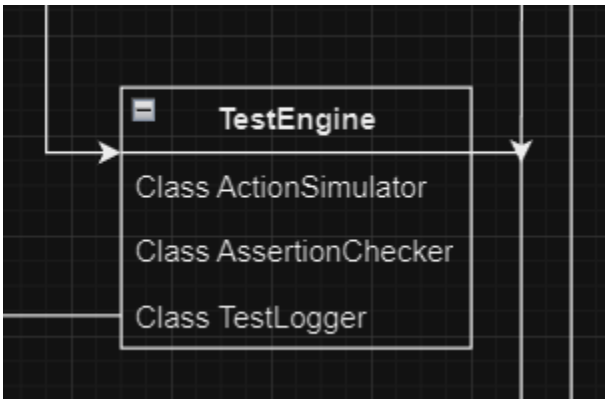


Рисунок 2.11 Модуль тестового двигуна

Модуль тестового двигуна: Основа фреймворку, яка відповідає за виконання тестових сценаріїв. Включає інструменти для імітації користувацьких дій, перевірки відповідності очікуваним результатам та логування процесу тестування.

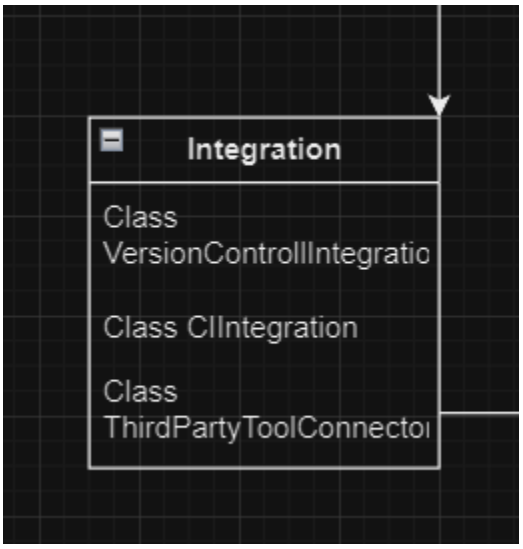


Рисунок 2.12 Модуль інтеграції з іншими інструментами

Модуль інтеграції з іншими інструментами: Цей модуль забезпечує інтеграцію з зовнішніми інструментами та сервісами, такими як системи контролю версій, інструменти неперервної інтеграції (CI/CD) та інші інструменти тестування.

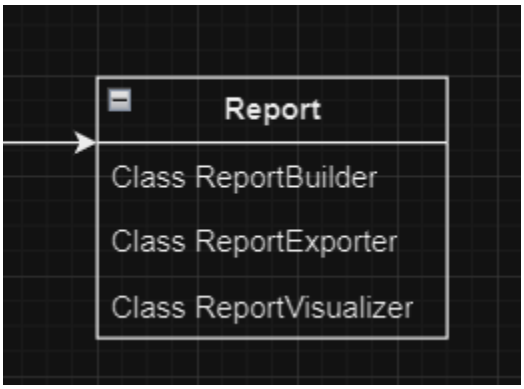


Рисунок 2.13 Модуль звітності

Модуль звітності: Відповідає за генерацію звітів за результатами тестування. Забезпечує формування детальних звітів, які можуть включати інформацію про успішність тестів, виявлені помилки, покриття коду тестами тощо.

Кожен з цих модулів виконує свою специфічну роль і разом вони формують комплексний фреймворк для автоматизованого тестування, який може бути адаптований до різних потреб користувачів і проектів.

Для кожного з описаних модулів фреймворку автоматизованого тестування веб-додатків можна визначити наступні класи та їх ролі:

Модуль інтерфейсу користувача (UI):

Клас TestRunner: Управління процесом запуску тестів.

Клас TestEditor: Редагування та створення нових тестових сценаріїв.

Клас ResultViewer: Відображення результатів тестування.

Модуль обробки даних:

Клас DataCollector: Збір даних під час тестування.

Клас DataAnalyzer: Аналіз зібраних даних для виявлення тенденцій та відхилень.

Клас ReportGenerator: Генерація звітів на основі аналізу даних.

Модуль комунікації з базами даних:

Клас DatabaseConnector: Встановлення з'єднання з базами даних.

Клас QueryExecutor: Виконання запитів до бази даних.

Клас DataMapper: Відображення даних бази на структури даних у програмі.

Модуль тестового двигуна:

Клас ActionSimulator: Імітація дій користувача у веб-додатку.

Клас AssertionChecker: Перевірка очікуваних умов та результатів.

Клас TestLogger: Запис подій та результатів тестування.

Модуль інтеграції з іншими інструментами:

Клас VersionControlIntegration: Інтеграція з системами контролю версій.

Клас CIIntegration: Інтеграція з інструментами неперервної інтеграції.

Клас ThirdPartyToolConnector: З'єднання з додатковими інструментами тестування.

Модуль звітності:

Клас ReportBuilder: Створення структури звіту.

Клас ReportExporter: Експорт звітів у різні формати.

Клас ReportVisualizer: Візуалізація даних у графічному вигляді.

Кожен клас визначає специфічні властивості та методи, які використовуються для виконання певних задач у рамках модуля. Разом ці класи формують комплексну архітектуру фреймворку, забезпечуючи необхідну функціональність для ефективного та гнучкого автоматизованого тестування веб-додатків.

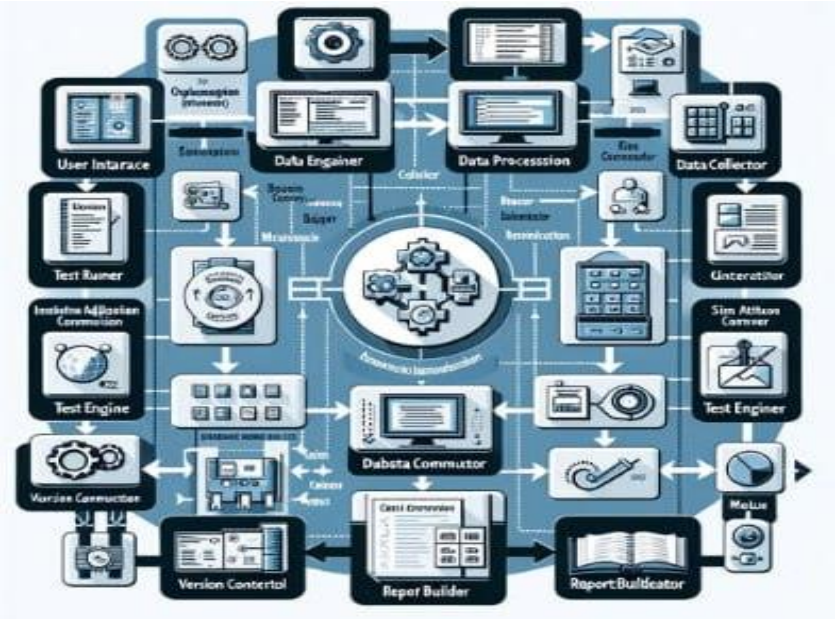


Рисунок 2.14. Графічне зображення модулів та класів

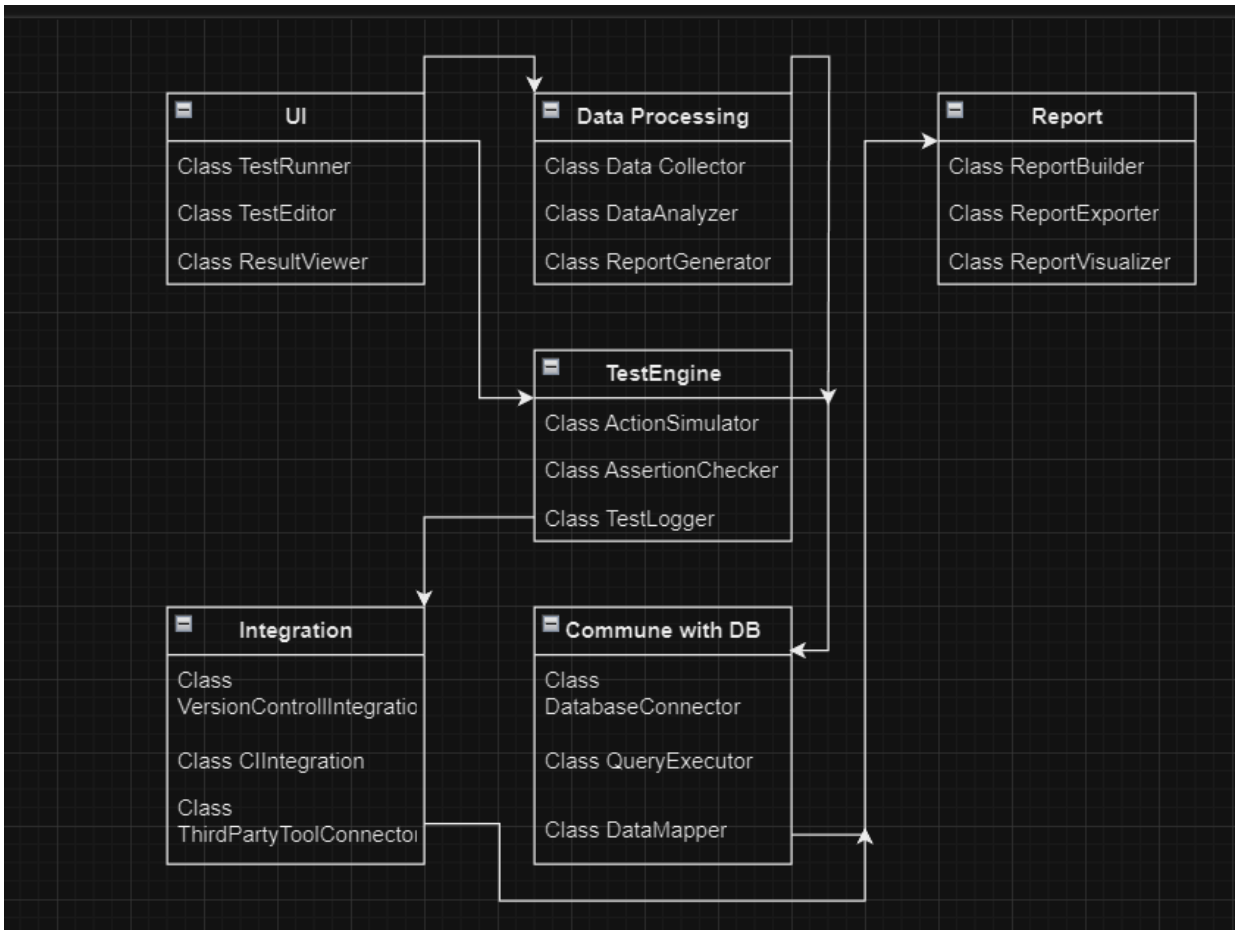


Рисунок 2.15 Діаграма взаємозв'язків модулів розробленого фреймворку

Автоматизація тестування може значно скоротити витрати компаній-розробників, заощадити час і ресурси на тестування, а також знизити ризики випуску на ринок неякісних продуктів. Тому технологія автоматизації тестування стає все більш популярною серед компаній, які займаються розробкою програмних продуктів. Автоматизоване тестування полягає у написанні автотестів – програм, вкладених у виконання закладених у яких тестових сценаріїв. Завдяки їм розробник може через короткий проміжок часу отримати інформацію про те, чи була виявлена помилка під час виконання цього сценарію, чи успішно пройшов тест. Для розробки та запуску автотестів використовують спеціальні системи автоматизованого тестування. Основна ідея автоматизованого тестування полягає в повній заміні людської праці. Програміст один раз створює програму, яка перевіряє всі підготовлені заздалегідь тестові сценарії. Далі сценарії лише підтримуються в

актуальному робочому стані. Створення таких програм може зменшити присутність ручного тестування в життєвому циклі програмного забезпечення [2].

Такий підхід економить час співробітників відділу контролю якості і значно прискорює роботу команди розробників і весь процес розробки ПЗ.

Недоліком автоматизації тестування є те, що вона застосовується лише для роботи з довгими проектами. Створення програми тестування займає багато часу і витрачати цей час на невеликі проекти просто безглуздо. Як правило, автоматизоване тестування ефективно застосовується до відносно простих сценаріїв та тестів. Автоматизація складних багатокрокових сценаріїв може зайняти неприпустимо багато часу через непередбачені нюанси або технічні проблеми [3]. Також може знадобитися багато часу для підтримки тестів, при необхідності часто вносити до них правки після змін у функціональності, що тестується. Незважаючи на нюанси автоматичного тестування, цей тип не поступається, і компанії впроваджують його у свій робочий процес.

У цій роботі розглядається розроблена авторами програмна система, призначена для побудови тестових сценаріїв та автоматизованого тестування за допомогою різних web-додатків.

Архітектуру системи описано відповідно до рекомендацій Rational Unified Process. Специфікація UML системи розділена на такі частини:

– Концептуальні моделі. Ця модель описує учасників системи, основні сценарії та варіанти використання системи;

– Логічна модель. Ця модель включає опис та діаграми найбільш важливих модулів системи, описи значущих класів, діаграми послідовностей виконання типових завдань та процесів додатку;

– Модель реалізації. Ця модель включає опис поділу прототипу системи на окремі компоненти, незалежні завдання, підпрограми, інформаційні та керуючі потоки та зв'язки між елементами системи.

На діаграмі варіантів використання показані учасники системи (актори) та основні варіанти роботи системи (прецеденти):

– Тестувальник створює/редагує сценарій тестування web-програми, використовуючи можливості фреймворку Selenium Web Driver [4] та середовища юніт-тестування NUnit для мови C#. Сценарій зберігається як виконуваного файла .exe щодо наступних тестів.

– Програміст запускає виконуваний файл і система проводить автоматичне тестування програми. Після завершення тестування генерується звіт, що містить інформацію про час початку та завершення тестування, кількість запущених тестів, відсоток успішно пройдених тестів.

Логічна модель включає опис та діаграми найбільш важливих модулів системи, описи значущих класів, діаграми послідовностей виконання типових завдань та процесів додатку.

Клас <Поточна сесія> містить змінні для підрахунку кількості пройдених та не пройдених тестів, адреси каталогів, а також методи ініціації та проведення функціонального тестування:

–змінна <IWebDriver> служить вказівником на інтерфейс, через котрий користувач керує браузером;

–змінні <WorkDir>, <FolderResult>, <FolderScreen>, <PathReportFile> містять інформацію про шляхи розташування компонентів програми;

–змінні <TestCountGood>, <TestCountFail> є лічильниками позитивного та негативного проходження тестів.

Клас <Поточний тест> містить атрибути, що надають інформацію про проходження тесту; ініціює та зберігає такі змінні як:

–змінна <TestNumCurrent> зберігає рядок з найменуванням тест-сценарія, що протікає.

–змінна <ExexTestGood> служить індикатором проходження тестування.

–змінна <Ex> є вказівником на повідомлення, що виникає у разі виявлення помилки.

Клас <Генераціязвіту> містить атрибути для створення звіту, ініціює та зберігає такі змінні як:

–змінна <Step> яка є ідентифікатором етапу внесення інформації: 0 – початок звіту; 1 – формування звіту; -1 – завершення звіту.

–змінна <TestName> зберігає рядок з найменуванням тест-сценарія, що протікає.

–змінна <Result> зберігає результат проходження тестування: True-пройдено, False-не пройдено.

–змінна <Mesage> зберігає рядок з повідомленням, що виводиться у разі виникнення помилки.

Модель реалізації описує поділ системи на окремі компоненти, незалежні завдання, підпрограми, інформаційні та керуючі потоки та зв'язки між елементами системи.

У розробленій програмній системі Visual Studio відбувається компіляція коду мовою C#, який передається до виконання утиліті NUnit.consolerunner.

Скрипт тесту містить команди, що імітують взаємодію користувача з браузером Google Chrome за допомогою драйвера Selenium WebDriver. При цьому всі внесені або змінені дані записуються в існуючу базу даних, пов'язану з веб-додатком, що тестується.

2.5 Технологічне забезпечення задачі

Автоматизоване тестування – це метод перевірки програмного забезпечення, за якого основні функції та кроки тесту, такі як запуск, ініціалізація, виконання, аналіз і вихід, виконуються автоматично за допомогою автоматизованих засобів тестування.процес. Програми можна тестувати на різних рівнях: код (модульні та інтеграційні тести), API та GUI. Залежно від ситуації більш доречними є різні типи тестів. Розглянемо види автоматизованих тестів докладніше.тестування.

Види автоматизованого тестування поділяються на:

1. Unit-тестування(або модульне):

Коректність класів і методів перевіряється розробником, що написав код, для

виявлення найбільш очевидних помилок під час процесу написання самого коду.

2. ТестуванняAPI:

REST/SOAP сервіси перевіряються розробниками послуг або фахівцями з автоматизації тестування для виявлення помилок під час етапу модульного тестування.

3. UI-тестування:

Графічний інтерфейс перевіряється спеціалістами з автоматизації тестування з метою поліпшення умов роботи ручних тестувальників та швидшого виявлення помилок на етапі приймального тестування.

Модульне тестування — це тестування модуля коду (як правило, функції або класу у випадку ООП-коду) в ізольованому середовищі. Це означає, що якщо в коді використовуються сторонні класи, замість них використовуються класи-заглушки (моки та стаби). В такому випадку код не повинен взаємодіяти з мережею (та зовнішніми серверами), файлами чи базою даних. В іншому випадку ми тестуємо не лише саму функцію або клас, а й його взаємодію з мережею, файлами, базою даних і так далі.

Стаб – це клас-заглушка, який повертає дані замість виконання дії. Наприклад, заглушка класу бази даних може повернути успішний запит замість фактичного доступу до бази даних. І коли ви намагаєтеся щось з нього прочитати, він повертає масив заздалегідь підготовлених даних.

Мок – це клас-заглушка, який використовується для перевірки, чи викликається певна функція.

Модульні тести зазвичай передають різні вхідні дані функції та перевіряють, чи повертає вона очікуваний результат. Наприклад, якщо у вас є функція, яка перевіряє точність номера телефону, ви можете надати йому попередньо визначений номер і перевірити, чи можна його правильно ідентифікувати. Якщо у вас є функція, яка розв’язує квадратне рівняння, перевірте, чи повертає вона

правильні корені (для цього необіхдно заздалегідь створити список рівнянь із відповідями).

Модульні тести підходять для тестування коду, який містить логіку. Написання модульних тестів може бути складним, якщо ваш код містить мало логіки та переважно посилання на інші класи.

API – це набір функцій, які можна викликати, щоб отримання даних. Наприклад, сервіс Гугл.Карти має свій API геокодера. Ви можете отримати координати точки, надіславши запит із географічною адресою (і навпаки). Центральні банки також мають API, які повертають офіційний курс валют у заданий день. Якщо ваша програма має API, ви можете перевірити її, надіславши підготовлений запит і порівнявши відповідь з очікуваною.

GUI – це графічний інтерфейс, тобто те, що користувач бачить на екрані. Це найскладніший для перевірки тип. Щоб побачити, як працює веб-сайт, він має імітувати деякі досить складні функції браузера та аналізувати інформацію, що відображається на сторінці. Однак цей тип тестування дуже важливий, оскільки він взаємодіє з вашою програмою так само як користувач. GUI тести ще називають End-to-End (E2E) або приймальні (aceptance) тести.

Зробимо проміжний висновок про те, що необхідно автоматизувати:

1. Важкодоступні області у системі (backend-процеси, лог-файли, записи у БД);
2. Функціональність, яка часто використовується та має високий ризик помилок. Автоматизуючи перевірку критичної функціональності, ми можемо гарантувати швидке виявлення помилок та їх ефективне усунення;
3. Рутинні операції, такі як перебори даних у формах з великою кількістю полів. Автоматизація заповнення полів різними даними та перевірка після збереження спрощує процес та покращує ефективність;
4. Валідаційні повідомлення (автоматизувати заповнення полів некоректними даними та перевірку на появу тієї чи іншої валідації);
5. Довгі End-to-End сценарії;

6. Перевірка даних, які потребують точних математичних розрахунків; Перевірка правильності пошуку даних.

Висновки до розділу

Будь-яке тестування вимагає змістовного підходу із застосуванням технік тест-аналізу та тест-дизайну. Інакше є ризик назавжди пропускати велику кількість помилок. А цінність праці у тестуванні буде мінімальною. Без розуміння методик та способів застосування тест-аналізу та тест-дизайну тестувати якісно ПЗ практично неможливо. В цілому, ключові положення тест-аналізу та тест-дизайну застосовні як до тестування десктопних додатків, так і до Інтернету. Однак, із суттєвим застереженням – необхідно враховувати всі згадані нюанси тестування веб-додатків.

3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

3.1 Формування вимог до системи

Враховуючи критерії для оцінки ПЗ для автоматизації тестування, що наведені в першому розділі, можна виділити наступні характеристики які повинен мати розроблений фреймворк:

- Написаний на поширеній мові для веб-розробки;
- Багатофункціональність;
- Мультиплатформеність;
- Гнучкість у використанні.

Для задоволення всіх вище вказаних пунктів доцільним є використання скриптової мови PHP. Багато вед-сайтів та веб-додатків побудовані з використанням PHP, тому фреймворк для автоматизованого тестування може бути легше інтегрувати в екосистему вже існуючих ведб-додатків. Також PHP надає зручні функції для роботи з HTTP запитами, що дозволяє легко взаємодіяти з веб-додатками під час процесу тестування. Це особливо важливо для фреймворків автоматизованого тестування, які часто повинні відправляти запит та аналізувати відповіді для перевірки правильності роботи додатку. Існує велике велика кількість різноманітних розширень та бібліотек, що дозволить леко адаптувати фреймворк під потреби кожного окремого проекту при їх наявності.

Велика кількість веб-додатків використовують різні системи управління базами даних (СУБД). PHP підтримує багато різних типів СУБД, таких як MySQL, PostgreSQL, SQLite та інші, що буде неодмінно корисно. PHP також добре інтегрується з різноманітними інструментами для автоматизації тестування, такими як PHPUnit, Codeception, Behat та інші. Це дозволить використовувати фреймворк для автоматизованого тестування в різних складних тестових сценаріях. І не варто забувати про досить велику і активну спільноту розробників, що в свою чергу означає більшу поширеність для фреймворку а також легкість в знаходжені

необхідної документації та відповідей на можливі питання для легшої роботи з фреймворком.

3.2 Вибір мови інструментальних засобів програмування

JavaScript – мова програмування, яка потрібна для керування сценаріями перегляду Web-сторінок. Одна з особливостей цієї мови полягає в тому, що він дає можливість змінювати властивості середовища відображення на веб-сайті без перезавантаження самої веб-сторінки сайту. Наприклад, завдяки мові JavaScript можна поміняти фон Web-сторінки або форму кнопки, зробити слайдер онлайн або вивести вікно повідомлень.

JavaScript є об'єктно-орієнтованою мовою програмування. Однак через використання прототипування, що дозволяє швидко реалізовувати базовий функціонал для аналізу системи, в ній виникає ряд особливостей, пов'язаних із непрямим виконанням коду мови. Це відрізняє її від традиційних об’єкто-орієнтованих мов програмування. Крім того, в JavaScript існують певні властивості, які спільні для інших функціональних мов, таких як розгляд об'єктів як списків, використання анонімних функції, і функцій як об'єктів першого класу [3, с. 848].

Основні відмінностями мови JavaScript від мови C включають такі аспекти:

- Використання об'єктів з можливістю самоаналізу;
- Функції, які є об'єктами першого класу;
- Автоматичне наведення типів;
- Можилвість використання анонімних функцій.

Серед недоліків мови JavaScript можна виокремити такі:

- Відсутність можливості контролю областей видимості;
- Відсутність будь-якого інтерфейсу;
- Залежність від стандартних інтерфейсів доступу до веб-серверів та баз даних.

Щоб визначити важливість та значущість мови JavaScript, розглянемо деякі області її застосування:

- Розробка веб-додатків, де JavaScript дозволяє встановлювати лічильники, передавати дані між формами та інтегрувати ігри на веб-сайт;
- "Активна участь" у технології AJAX, яка суттєво прискорює взаємодію додатків, обмінюючи дані із сервером у фоновому режимі;
- Використання в операційних системах;
- Розробка мобільних додатків;
- Сфера навчання, де JavaScript є необхідним елементом у вивченні програмування в університетах для будь-якої програмістської спеціальності.

Каскадні таблиці стилів або CSS – це формальна мова опису зовнішнього вигляду Web-сторінки, написана за допомогою мови розмітки. Без CSS неможливе написання будь-якого Web-сайту, адже він задає стилі всім видимим елементам сторінки. За допомогою CSS встановлюють шрифт, розмір та колір тексту, розташування блоків та їх форму, фонове зображення сайту та адаптацію сайту до всіх можливих розширень девайсів [5, с. 272]. Основним завданням, що вирішується під час використання CSS, є зниження складності структури вмісту шляхом поділу та позначення елементів. Крім цього, за допомогою CSS можна представити ту саму Web-сторінку в різних стилях відображення, наприклад, екранне подання, читання голосом і т.д. Сам CSS складається з ряду правил, кожне з яких включає один або кілька перемикачів, які з'єднані комами, і блок значень, взятий у фігурні дужки ({}) і що складається з властивостей і значень [4, с. 414].

Під час відображення Web-сторінки її елементи можуть бути прив'язані до різних стилів, які мають певну ієрархію:

- авторські стилі (стилі, що надаються розробниками сторінки);
- стилі користувача;
- стиль браузера.

Але цю ієрархію можна перебити, додавши до будь-якої властивості стилю слово “! important” наприкінці перед знаком, що розділяє (;).

Крім цього, каскадні таблиці стилів дозволяють працювати зі шрифтовим оформленням Web-сторінки на вищому рівні, на відміну від стандартного HTML-коду, без обтяження сторінок графікою.

3.3 Розробка архітектури системи автоматизованого тестування

Розглядаючи процес розробки Web-програми як процес написання коду в текстовому документі, під середовищем розробки розумітимемо найбільш зручний і функціональний текстовий редактор. Через те, що у звичайному "Блокноті" від Windows хоч і можна написати код сайту, це буде вкрай незручно і довго, оскільки в "Блокноті" немає спеціальних плагінів, що спрощують роботу з кодом. Тому розглянемо модернізованіші аналоги, такі як SublimeText, PHPStorm, Atom. Кожен з них має мінімальні необхідні властивості, які значно спрощують роботу з текстом і сильно збільшують швидкість написання коду [5, с. 370].

Складемо список деяких важливих властивостей:

- приємний, зрозумілий та мінімалістичний інтерфейс;
- наявність міні-картки коду для зручного переміщення;
- можливість роботи з кодом у 2 та більше вікна;
- можливість структурувати код за допомогою комбінації клавіш;
- очевидні доповнення коду та завершальних тегів;
- величезна кількість плагінів під будь-які потреби.

Тепер для наочності порівняємо роботу з кодом у додатку «Блокнот» (рисунок 3.1) та у додатку «Sublime Text 3» (малюнок 3.2) – різниця колосальна, переваги очевидні.

Тепер розберемося докладніше на відмінностях вже спеціалізованих вищеперелічених текстових редакторів між собою. PhpStorm є

інтелектуальним текстовим редактором для PHP, HTML і JavaScript з можливістю аналізу коду на льоту і здатністю виправлення та запобігання помилкам у коді.

Рисунок 3.1. Демонстрація відображення коду у програмі «Блокнот»

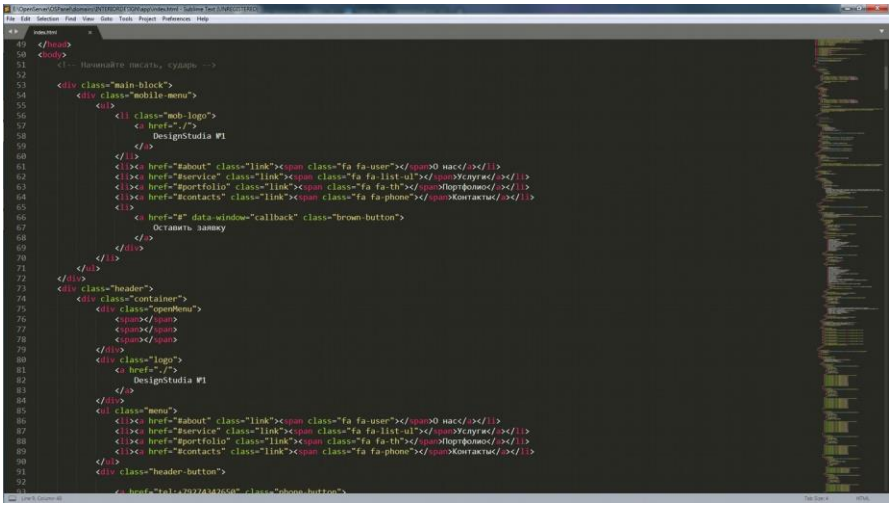


Рисунок 3.2. Відображення коду у програмі «Sublime Text 3»

Тепер розберемося докладніше на відмінностях вже спеціалізованих вищеперелічених текстових редакторів між собою. PhpStorm є інтелектуальним текстовим редактором (рисунок 3.3) для PHP, HTML та JavaScript з можливістю аналізу коду на льоту та здатністю виправлення та запобігання помилкам у коді.

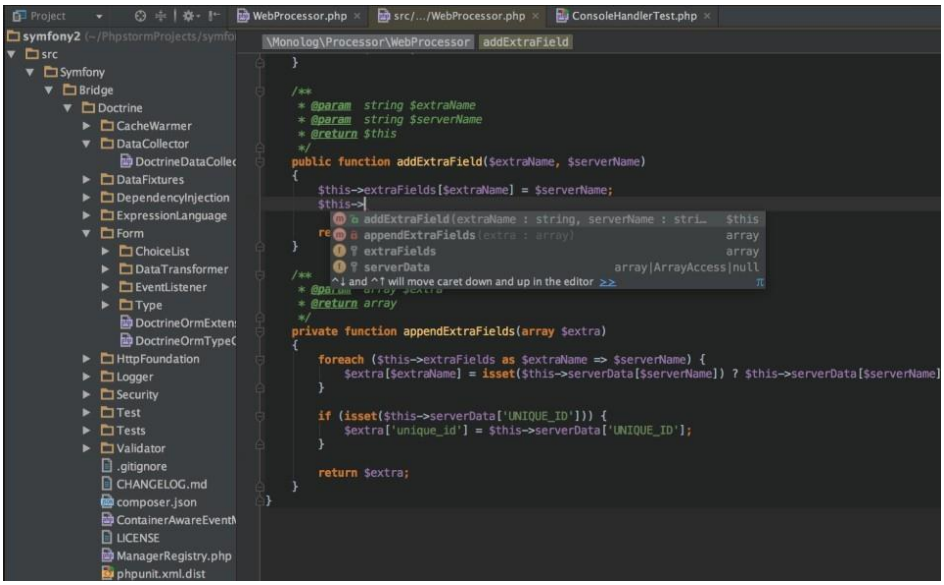


Рисунок 3.3. Зовнішній вигляд редактора PhpStorm.

Даний редактор нам не підходить, оскільки найчастіше він використовується для розробки об'ємних Web-проектів, тому має величезний внутрішній функціонал, який створює велике навантаження на комп'ютер. У разі слабких характеристик комп'ютера це може викликати лаги або довге завантаження. Порівняємо два текстові редактори «Atom» (рисунок 3.4) і «Sublime Text 3» (рисунок 3.2).

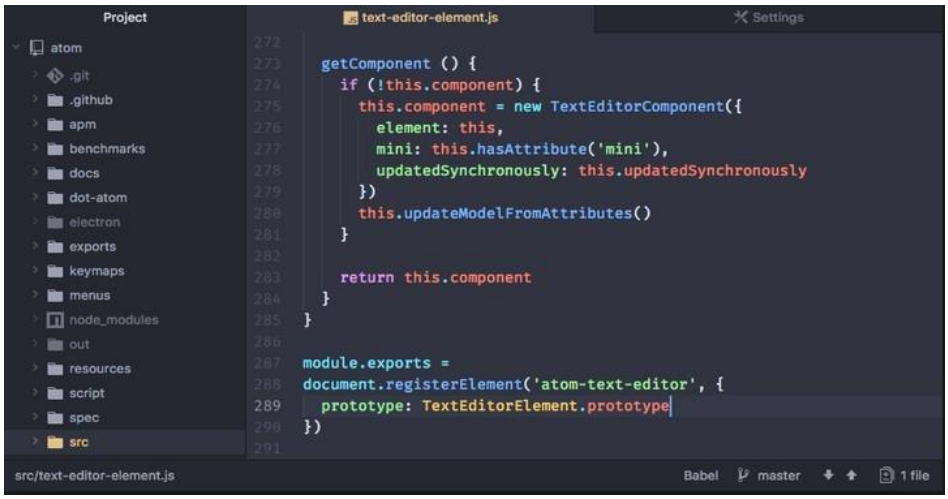


Рисунок 3.4. Зовнішній вигляд редактора Atom.

Основними відмінностями між ними є:

- у Sublime Text 3 більша продуктивність;
- налаштування зручніші в Atom;
- підтримка синтаксису в Atom гірше;

- в Sublime Text 3 є більше популярних плагінів для роботи з кодом.

З усього перерахованого вище робимо висновок, що доцільніше використовувати Sublime Text 3.

3.4. Розробка системи автоматизованого тестування

Існують спеціальні інструменти для тестування веб-інтерфейсу додатків – валідатори. Валідатор – це програма, яка перевіряє відповідність будь-якого документа, потоку даних або фрагмента коду певному формату. У випадку відсутності достатніх знань, для оцінки відповідності верстки стандартам, можна використовувати автоматичні засоби. Аналізуючи результати, можна вказати розробникам на найбільш серйозні помилки. Важливо пам’ятати, що іноді валідатори можуть акцентувати увагу на найменш суттєвих помилках, які, ймовірно, не потребують виправлення або не варті того зі сторони ефективної витрати ресурсів.

Якщо ви зафіксували такі невеликі недоліки і надсилаєте звіти про помилки, корисно буде зібрати їх у єдиний документ та долучати до відповідного звіту. До таких дрібних багів можна віднести всілякі рекомендації, які не впливають на відображення та функціонування контенту. Якщо подивитися перший приклад, то тег img не має alt-атрибута. Alt – це параметр, який каже, який текст буде писатися, якщо у вас, наприклад, відключені зображення, або зображення не завантажилося через повільне Інтернет-з'єднання.

Це потрібно, наприклад, якщо люди мають проблему із зором. Люди, які мають обмеження зору, часто використовують спеціальні програми – скринрідери.

Скрінрідери дозволяють визначати та інтерпретувати події, що відбуваються на екрані. Якщо скринрідер читає по екрану, то він не може рахувати картинку, а лише alt-текст. І якщо цей текст не заданий, людина ніколи не дізнається, що там було.

Однією з важливих складових веб-застосунків є форми для заповнення,

взаємодія з якими користувач здійснює за допомогою клієнта. Проте, дані форми найчастіше є джерелом дефектів, які можуть завдати великих фінансових і репутаційних збитків компанії.

Щоб не пропустити помилки у формах на продуктивне середовище, необхідно:

1. Ретельно перевірити обов'язковість заповнення полів та наявність відповідного маркування у них;

2. Після того, як форму заповнено та відправлено, необхідно переконатися в тому, що з даними відбувається саме те, що очікується. Наприклад, якщо дані мають бути внесені до бази даних (БД), необхідно перевірити, чи коректно заповнилися всі поля у потрібних таблицях. Для цього необхідно мати доступ до БД, до схем та таблиць, а також знання структурованої мови запитів (SQL);

3. Використовувати чит-листи для тестування форм. Чит-листи – набір перевірок, що повторюються. Чит-листи складаються з їх подальшого багаторазового використання. Наприклад, ч- листів може бути введення можливих і неможливих значень типів даних, спеціальних символів, символів в іншому кодуванні;

4. Необхідно перевірити, чи виводяться зрозумілі користувачеві інформаційні повідомлення необхідність заповнення порожніх полів після спроби надіслати форму;

5. Звертати увагу на реалізацію екранування символів у полях форм, що є потенційним джерелом вразливостей для програми та користувачів. Екранування повинно здійснюватися не лише на рівні клієнта, але й на рівні сервера, відключити який у клієнті досить просто (наприклад, за допомогою спеціальних плагінів, що знімають усі можливі обмеження у декілька натискань, таких як Web Developer Toolbar – Forms);

6. Переконатися, що після заповнення форми користувач отримує лист із підтвердженням, в якому зазначено відправника, а тіло листа відповідає вимогам,

таким як наявність працездатних посилань;

7. Використовувати спеціальні допоміжні інструменти для тестування форм такі як Web Developer Toolbar.

Мережа Інтернет має величезну цінність завдяки своєму практично нескінченному обсягу інформації. Багато цієї інформації подається у вигляді текстів по ходу того як користувач взаємодіє з застосуванням клієнта. Більшість веб-ресурсів у певному розумінні вимагають перевірки текстів на наявність граматичних та друкарських помилок.

Хоча значущість цього тестування може бути не такою великою порівняно із функціональним тестуванням, його важливість не повинна бути недооціненою. Існують програми для перевірки правопису, які можна використовувати онлайн або як плагіни для браузерів. Тестування частини веб-програми, що розміщена на веб-сервері, можливе без використання графічного (клієнтського) інтерфейсу, хоча це вимагає від фахівця певного рівня технічних знань і використання додаткових інструментів. Усі компоненти веб-програми повинні взаємодіяти між собою, і це здійснюється завдяки протоколу HTTP(s). Без HTTP наша багатостороння система не могла б функціонувати взагалі, оскільки HTTP є ключовим протоколом передачі даних у нашій клієнт-серверній архітектурі.

Взаємодія відбувається за допомогою повідомлень, що включають запити та відповіді: сервер повинен відповісти на надісланий запит від клієнта..

При взаємодії з відповідями спеціаліст із тестування перш за все повинен звертати увагу на методи та коди стану, які містяться у початковому рядку. Серед найбільш популярних методів, які були описані у першому розділі, можна виділити GET та POST. Існують і інші методи, такі як PUT, DELETE, CONNECT і т. д., але вони використовуються менш часто, ніж GET та POST. Для створення запитів часто використовуються такі програми, як Fiddler або Postman. Вони дозволяють легко відстежувати всі запити від клієнта та відповіді, переглядати їх деталі, а також вносити зміни та відправляти модифіковані запити на сервер для оцінки поведінки системи у таких ситуаціях..

При тестуванні запитів необхідно:

1. Розуміти, чи правильний метод використовується для того чи іншого запиту;
2. Розуміти запити. Розшифровувати запити досить просто, особливо якщо це запит типу GET - навіть для пересічного користувача вони логічно зрозумілі. Аналіз запитів надає можливість виявити дефекти набагато швидше, ніж при їх пошуку в інтерфейсі.;
3. Відстежувати трафік щодо запитів на інші сервери, які до вас не належать;

Слідкуючи за трафіком, необхідно дуже уважно стежити за кодами станів.

Висновки до розділу

Кожна відповідь на запит несе в собі значну кількість корисної інформації для розробників та фахівців з тестування програмного забезпечення. Одне із важливих джерел такої інформації - це код стану. За цим кодом клієнт отримує результати свого запиту та визначає наступні дії. Набір кодів стану є стандартом, який описаний у відповідних документах RFC (Request for Comments). Кожного разу, коли потрібно створити звіт про непрацюючі посилання, інші елементи веб-сторінки або системні помилки, слід обов'язково вказувати відповіді сервера. Це допоможе зекономити час розробників у визначенні причин дефекту.

4. ТЕСТУВАННЯ ДОДАТКУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Тестування додатку

Система побудована на основі NUnit-фреймворку, призначеного для модульного тестування. NUnit підтримує використання атрибутів, необхідні розпізнавання тестів.

Атрибут [TestFixture] означає, що всі позначені ним класи утворюють тестову структуру.

Атрибут [OneTimeSetUp] означає, що позначені ним класи викликатимуться перед початком тестування.

Атрибут [OneTimeTearDown] означає, що позначені ним класи будуть викликатися одного разу після закінчення тестування.

Атрибут [SetUp] означає, що позначені ним класи будуть викликатися перед початком кожного тесту.

Атрибут [TearDown] означає, що позначені ним класи будуть викликатись після закінчення кожного тесту.

Атрибут [Test] означає, що всі помічені ними класи є безпосередньо тестами.

Таким чином, проект є такою структурою:

```
[TestFixture]
classTest
{       public static voidMain(string[] args) {}
                [OneTimeSetUp]public voidTestFixtureSetUp(){}
                [OneTimeTearDown]public voidTestFixtureTearDown(){}
                [SetUp]public voidSetUp(){}
                [TearDown]public voidTearDown(){}
                [Test]public voidTEST_1(){}
}
```

Лістинг 1. Структура проекту.

Для коректного запуску тестів за допомогою файлу будемо використовувати утиліту nunit3-console.exe, яка підключається до проекту за допомогою вбудованого в VisualStudio менеджера пакетів NuGet. Функція запуску тестів має вигляд:

```
        public static voidMain(string[] args)
{
        Stringpath=Assembly.GetExecutingAssembly().Location;NUnit.ConsoleRunner.
Program.Main(new[] {path});
}
```

Лістинг 2. Функція запуску тестів.

Метод <GetExecutingAssembly().Location> отримує шлях до розташування завантажувального файлу, а метод Program.Main, що належить простору імен <NUnit.ConsoleRunner>, викликає утиліту nunit3-console.exe якої передається шлях до скомпільованого файлу.

Конструювання тест-кейсів. Побудова тест-кейсів починається з

```
public stringSname = Randomizer.CreateRandomizer().GetString(8);public stringSlogin = Randomizer.CreateRandomizer().GetString(10);public stringSphone = Randomizer.CreateRandomizer().GetString(9,"1234567890");public stringSpassword = Randomizer.CreateRandomizer().GetString(6);public
```

оголошення глобальних змінних, що використовуються:

Лістинг 3. Оголошення глобальних змінних.

Змінні <Sname>, <Slogin>, <Spassword>, <Sphone> зберігають випадково згенеровану інформацію про користувача сайту, яка надалі використовуватиметься для доступу до програми.

Тип класу <TestFixtureSetUp()> представлений нижче на лістингу 4.

<TestFixtureSetUp()> має атрибут [OneTimeSetUp], що означає, що він буде викликатись перед стартом тестування. За допомогою методу <CreateDirectory()> створюються нові каталоги, в яких зберігатимуться звіт та скріншоти, а метод <GenerateReport()> викликає функцію створення звіту.

Визначаємо <driver> як покажчик змінної типу <ChromeDriver>, що належить пакету <Selenium.WebDriver.ChromeDriver>. Це дозволить нам керувати діями браузера Google Chrome.

Метод <Manage()> дозволяє керувати налаштуваннями драйвера, з його допомогою встановлюється роздільна здатність вікна програми "На весь екран" і час очікування до появи наступного елемента в 5 секунд.

```
[OneTimeSetUp]
public voidTestFixtureSetUp()
{
if(Directory.Exists(iFolderResultTest)) Directory.Delete(iFolderResultTest,true);
DirectoryInfo dr = Directory.CreateDirectory(iFolderResultTest);
DirectoryInfo ds = Directory.CreateDirectory(iFolderScreen);
GenerateReport(0,"0",true);
ChromeOptions options
=newChromeOptions(); options.AddArguments("--
ignore-certificate-errors"); options.AddArguments("-
-ignore-ssl-errors");
driver =newChromeDriver(iWorkDir, options);
driver.Manage().Window.Maximize();
driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(5);
}
```

Лістинг 4. Клас <TestFixtureSetUp()>.

Клас <SetUp> має вигляд:

```
[SetUp]
public voidSetUp()
{ driver.Navigate().GoToUrl("http://localhost/index.php"); }
```

Лістинг 5. Клас <SetUp>.

За допомогою методу <Navigate()> браузер переходить на сторінку тестованого додатка перед запуском кожного тесту.

Об'єктом для проведення тестування було обрано web-додаток, створений студенткою, яка навчається за напрямом підготовки «Програмна інженерія», та реалізує косметику за допомогою мережі Інтернет.

4.2. Аналіз результатів тестування

Було створено 10 тест-кейсів, які перевіряють працездатність основного функціоналу web-додатку:

–тест-кейс №1 – Перевірка проходження реєстрації,

–тест-кейс №2 – Перевірка проходження авторизації,

–тест-кейс №3 – Перевірка правильності відображення меню,

–тест-кейс №4 – Перевірка правильності відображення товарів,

–тест-кейс №5 – Замовлення товару,

–тест кейс №6 – Пошук товару,

–тест-кейс №7 – Перевірка купівлі товару,

–тест-кейс №8 – Перевірка товару на складі,

–тест-кейс №9 – Редагування ціни товару адміністратором,

–тест-кейс №10 – Перевірка сумісності із браузером Internet Explorer.

Докладно розглянемо створення тестового сценарію для тест-кейсу №1 – Перевірка реєстрації.

Пакет Selenium.WebDriver надає такі методи взаємодії з вмістом web-сторінки:

Метод <FindElement()> знаходить насторінці перший елемент, який відповідає умовам пошуку в залежності від обраного методу:

– By.Id() – пошук елемента відбувається за найменуванням ідентифікуючого його поля Id;

– By.Name() – пошук елемента відбувається за найменуванням його поля

Name;

– By.Xpath() – пошук елемента відбувається за допомогою мови запитів до елементів XML-документу XPATH

Метод <Click()> клацає лівою кнопкою миші по вибраному елементу. Метод <SendKeys()> здійснює введення тексту у вибраний елемент.

Код тест-сценарію №1 має вигляд:

```
[Test]
public voidTEST_1()
{iTestName ="registration";try
{driver.FindElement(By.Id("login")).Click();
driver.FindElement(By.Id("registration")).Click();
driver.FindElement(By.Name("FIO")).SendKeys(Sname);
driver.FindElement(By.Name("login")).SendKeys(Slogin);
driver.FindElement(By.Name("password")).SendKeys(Spassword);
driver.FindElement(By.Name("phone")).SendKeys(Sphone);
driver.FindElement(By.XPath("//button[@type='submit']")).Click();Assert.IsTrue(driver.
FindElement(By.XPath("//*[contains(text(),'Ви успішно зареєстровані!')]")).Displayed);
GenerateReport(1, iTestName,true); iExecTestGood =true; }catch(Exception ex)
{ GenerateReport(1, iTestName,false, ex.ToString()); iExecTestGood =false;}
Assert.True(iExecTestGood); }
```

Лістинг 6. Код тест-сценарію №1.

В результаті проведення автоматизованого тестування за допомогою розробленої авторами програмної системи було отримано звіт, представлений на рисунку 4.1. На рисунку видно, що «провалено» 4 сценарії: Пошук товару, Перевірка купівлі товару, Перевірка товару на складі та Перевірка сумісності з браузером Internet Explorer :

Рисунок. 4.1. Звіт про проходження тестування.

Завдяки зробленим при виникненні помилки знімкам екрана було виявлено помилки, зроблені розробником під час написання коду web-програми.

Висновки до розділу

При виборі параметрів та значень потрібно пам'ятати, що негативні тести не варто включати до таблиці – в одному тесті може перевірятись кілька пар, а у разі негативного тесту буде виконано перевірку лише одного параметра, в результаті деякі пари можуть залишитись неперевіреними. З цієї причини в даному прикладі відсутні значення обсягу даних, що рівні нулю і перевищують об'єм диска. Якщо їх додати, то в результаті використання методу може вийти кейс, в якому на нульовому обсязі даних буде перевірятися наприклад пара файлової системи ISO і початку мультисесії. В результаті, успішно переконавшись у коректній обробці спроби запису порожнього диска, ми пропустимо перевірку пари ISO-почати мультисесію.

ВИСНОВКИ

На сучасний день веб-застосунки продовжують активно розвиватися, а розмаїття використовуваних технологій та різноманіття дефектів лише збільшуються.

В даній роботі проведено аналіз фреймворків для автоматизованого тестування веб-додатків, а також технік і методологій їх тестування на різних етапах життєвого циклу розробки програмного забезпечення. В результаті було досліджено основи взаємодії клієнтської та серверної частин веб-додатків, структуру веб-сторінок, аспекти пошуку веб-елементів та методи їх тестування.

Деякі ключові висновки включають те, що функціональне (ручне) тестування проявляє свою ефективність при розробці нових бізнес-процесів чи проектів, у той час як автоматизоване тестування більш доцільне для перевірки існуючого функціоналу. Засновано на аналізі документації та методів тестування, були розроблені набори тестових випадків, спрямованих на перевірку працездатності нового, значущого бізнес-процесу отримання онлайн-пропозицій від клієнтів.

Додатково було проведено дослідження сутності автоматизованого тестування веб-додатків, і в результаті були розроблені автоматизовані сценарії тестування для онлайн магазину. Це призвело до позитивного економічного ефекту, значущого скорочення часу та витрат на ручне тестування, а також відчутного зменшення впливу людського фактора на процес тестування. Зусилля були направлені на усунення помилок з метою забезпечення якості програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Биков В.Ю., Лапінський В.В. Методологічні та методичні основи створення і використовування електронних засобів навчального призначення — 2012. С. 3-6.. 3.

2. Адаменко О.В., Духовна М.М., Панченко Л.Ф. та ін. Тестові завдання для контролю знань в курсі "Обчислювальна техніка і технічні засоби навчання": Навч.-метод, посібник / За ред. Т.О. Козлакової. — К.: ВІПОЛ, 1996.-84 с.

3. Алексейчук І.С. Про технологію створення системи тестування / І.С. Алексейчук // Нові технології навчання: Науково-методичний збірник. — К.: НМЦВД, 2000. — С.43-92.

4. What is REST? [Електронний ресурс]. — Режим доступу: http://www.restapitutorial.com/lessons/whatisrest.html. 8. Markus Egger — MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. — 2012. — Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprisearchitectures-silverlight-and-wpf.

5. Кейт Джонс. DOM Scripting: Web Design with JavaScript and the Document Object Model. / К. Джонс — Перше, 2005. — 368 с. 52

6. Anthony Gore — Full-Stack Vue.js 2 and Laravel 5 [Електронний ресурс]. — 2015. — Режим доступу: https://bit.ly/2OEODzR.

7. Martin Fowler — GUI Architectures. Часть 1 [Електронний ресурс]. — 2009. — Режим доступу: https://bit.ly/2CvCk1e.

8. Vue.js Material Component Framework — Vuetify.js [Електронний ресурс]. — 2016. — Режим доступу: https://vuetifyjs.com.

9. Л. Аткінсон, З. Сураскін. PHP5. Бібліотека професіоналу. / Л. Аткінсон, З. Сураскін: «Вільямс», 2006 — 543 с.

10. Скотт Хокінс. Адміністрування веб-сервера Apache і керівництво по електронній комерції. / С. Хокін: «Вільямс», 2001. — 336 с.

11. Болье А. — Learning SQL [Електронний ресурс]. — 2005. — Режим

доступу: http://shop.oreilly.com/product/9780596007270.do.

12. MySQL. Довідник. MySQL AB:«Вільямс», 2006 — 521 с.

13. Варган О. І., Павловський В. І., Аналіз проблеми автоматизованого тестування веб-додатків. Прикладна математика та комп'ютинг. ПМК, 2023 : шістнадцята наук. конф. магістрантів та аспірантів, Київ, 28—30 лист. 2023 р. : зб. тез доп. / [редкол.: Дичка І. А. та ін.]. — К. : Просвіта, 2023.

14. Варган О. І., Павловський В. І., Автоматизація тестування Android-додатків з використанням протоколу Webdriver на платфорі .NET. СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ, 2023 : п’ята наук. прак. конф. молодих вчених та студентів, Херсон, 30 лист. 2023 р.

ДОДАТОК 1

```
<! DOCTYPE html>
<html lang="en">
<head>
    <metahttp-equiv="content-type"content="text/html;  charset=utf-8" /
>
    <title>MapEditor GS</title>
    <link rel="shortcut icon" type="image/png"
href="/statics/favicon/faviconRekod.png"/>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7 x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="//cdn.jsdelivr.net/npm/slick-
carousel@1.8.1/slick/slick.css"/>< /span>
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/ css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr "
crossorigin="anonymous">
    <link rel="stylesheet" href="./css/fonts.css">
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" href="./css/media.css">

    <script src="js/jquery-1.7.2.min.js" type="text/javascript"></script>
    <script src="js/jquery-ui-1.9.2.custom.min.js"
type="text/javascript"></script>
    <script src="js/jwplayer/jwplayer.js" type="text/javascript"></script>
    <script src="js/jwplayer/jwplayer.html5.js"
type="text/javascript"></script>
    <script src="js/fancybox/jquery.fancybox.pack.js"type="text/javascript"></script>
    <script src="js/fancybox/jquery.mousewheel-3.0.6.pack.js"
type="text/javascript"></script>
    <scriptsrc="js/fancybox/jquery.fancybox-thumbs.js"
type="text/javascript"></script>

    <script  src="http://network.geocrm.ru/public/javascripts/geoportal/geoportal-
api.min.js"type="text/javascript"></script>
    <script src="js/app/HashMap.js" type="text/javascript"></script>
```

```
                    <script src="js/app/ModelEis.js" type="text/javascript"></script>
                    <script src="js/app/AFeatures.js" type="text/javascript"></script>
                    <script src="js/app/WMSFeatures.js" type="text/javascript"></script>
                    <script src="js/app/WFSFeatures.js" type="text/javascript"></script>
                    <script src="js/app/LeftPanel.js" type="text/javascript"></script>
                    <script src="js/app/LayerFeaturesBox.js" type="text/javascript"></script>


                    <scriptsrc="https://code.jquery.com/jquery-3.3.1.min.js"></script>
            <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"  integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0          W1"
crossorigin="anonymous"></script>
            <script          type="text/javascript"          src="//cdn.jsdelivr.net/npm/slick-
carousel@1.8.1/slick/slick.min.js"> </script>
                    <script src="./js/main.js"></script>
                    <scriptsrc="./js/jcore.js"></script>
                    <script src="./js/jquery.mobile.custom.min.js"></script>
                    <script src="./js/mask.js"></script>
                    <script src="./js/wow.js"></script>

            </head>
            <body>
                    <header class="header">
                            <div class="header-wrap">
                                    <div class="header-left">
                                            <div class="header-title">
                                                    <a href="./"><span>MapEditor</span></a>
                                            </div>
                                            <div class="header-controls">
                                                    <ul>
                                                    <li><a      href="#"><img      title='Виміряти
відстань' src="./img/distance.png" alt=""></a></li>
                                                    <li><a href="#"><img title='Друк карти'
src="./img/print.png" alt=""></a></li>
                                                    <li><a      href="#"><img      title='Список
об'єктів області' src="./img/bbox-search.png" alt=""></a></li>
```

```
        шар' src="./img/create-layer.png" alt=""></a></li>
                                                <li><a href="#"><img title='Фіксоване
посилання' src="./img/fixedLink.png" alt=""></a></li>
                                                </ul>
                                            </div>
                                        </div>
                                        <div class="header-right">
                                            <div class="header-admin">
                                                <a href="./admin.html">
                                                    <span class="admin-wrap">
                                                        <img src="./img/admin-icon.png"
        a
lt="">                                                  <span>Управління</span>
                                                    </span>
                                                <
                                                /a>
                                                </div
                                        >

                                            <div class="header-username">
                                                <span>
                                                    Здравствуйте, NVAnisimov
                                                </span>
                                            </div>
                                            <div class="header-exit">
                                                <a href="./reg-page.html">
                                                    <div class="exit-wrap">
        a                                               <img src="./img/exit-logo.png"
lt="">
                                                        <span>Вихід</span>
                                                    </div>
                                                </a>
                                            </div>
                                        </div>
                                    </div>
                                </header>
```

```
<script>
    $(function() {
        var application;

        GeoPortal.on(&ready",geoportalReady,this);fu
    nction geoportalReady() {
        application = new Application("depadmin",
"12345678", "-932516122", 50.745849609375, 55.466399363938194, 7)
            application.initialization();
        }


        функція Application(login, password, baseLayerId, lon, lat,
    zo
om) {
            var self = this;
        this.mapObject = null;
            this.layersStore = новий
        GeoPortal.HashMap(); this.marker = null;
            this.clickLatLng =
        null; this.featuresStore = null;
        this.leftpanel = null;
        this.featureWidget = null;

            this.initialization = function() {
                var baseLayer = findBaseLayer(baseLayerId),
            mapReady = function() {
                    self.mapObject.setZoom(zoom);
                self.mapObject.setCenter(lon,lat);
            };

                this.mapObject = new GeoPortal.Map('map',
    {baseLayer:
baseLayer});
            this.mapObject.on("ready", mapReady, this)

            GeoPortal.authenticate(login, password,
                function(data) {
                    GeoPortal.requestGroups(true,function(gro
                        ups) {
                            var len =
    groups.length, i=0;
                            for(i=0;i<len;i++) {
```

```
                                                                    }

                                                                        $
        ("#groups").find(".layer").find("input").on("click",function()      {
                                                                            var
(this).val(),
                                                                        id = $layer
        self.layersStore.get(id);
                                                                        =




                                                                            if(typeof
        'undefined') {                                                  layer !=

                layer.turn(self.mapObje
        ct);
                                                                            }
                                                                        });

                                                                    },

                                                                function(status,error)              {
                                                                        console.log("Error to
        request layers groups. Status = " + status + ". Error text: " + error);
                                                                        }
                                                                        );
                                                                function(){
                                                                        controls();

                                                                    },
                                                                function(status,error){
                                                                        console.log("Error to request
        authentication. Status = " + status + ". Error text: " + error);
                                                                        }
                                                                        );


                                                                        self.mapObject.on("popupclose",
        removeMarker, this);


                                                                self.mapObject.on("click",function(e)           {
                                                                        self.clickLatLng = e.latlng;
                                                                        }, this );

                                                                self.mapObject.on("featureClicked",function(data) {
```

```
                self.mapObject.off("popupclose",
        removeMarker,
self);                                                  removeMarker();
                                                self.mapObject.on("popupclose",

        removeMarker,
self);


                                                if(self.leftpanel != null) {
                                                        self.leftpanel.destroy();
                                                        self.leftpanel = null;
                                                }

                                                if (data.features == undefined) {
                                                        console.log("Request features

        error. Status = " + status + ". Error text: " + error);
                                                                return;
                                                }

                                                if(data.features.lenght == 0) {
                                                        alert("У точці нічого не
        знайде
но!");                                                          return;

                                                }

                                                        var i = 0, len =
                                                data.features.length;
        GeoPortal.HashMa                        self.featuresStore=new
p();
                                                        for(i=0;i<len;i++) {
                                                                var feature = data.features
                                                        [i];

                self.featuresStore.add(feature.id(),feature);


                                                        }

                                                        if(self.featuresStore.getCount() ==
        1) {
                                                                var feature =
                                                        data.features[0], layer =
        self.layersStore.get(feature.layerId());
```

```
        s
elf});                                                                  } else {
                                                                                self.featureWidget =
        new GeoPortal.Widget.WMSFeatures(null, {latLng: self.clickLatLng, application: self});
                                                                        }
                                                                } else {
                                                                        self.featureWidget      =      новий
GeoPortal.Widget.WMSFeatures(null, {latLng: self.clickLatLng, application: self});
                                                                        }
                                                                );


                                        };

                                    function  findBaseLayer(baseLayerId)  {  var
                                            baseLayer = null,
                                                    i=0, len, layer;

                                                    len = GeoPortal.baseLayers.schemas.length;
                                            for(i=0;i<len;i++) {
                                                        layer                                       =
                                                    GeoPortal.baseLayers.schemas[i];    if(layer.id()
                                                    == baseLayerId) {
                                                                baseLayer    =
                                                        layer; break;
                                                    }
                                                }
                                                if(baseLayer == null) {
                                                        len = GeoPortal.baseLayers.spaces.length;
                                                    for(i=0;i<len;i++) {
                                                                layer =
        GeoPortal.baseLayers.spaces[
i];                                                             if(layer.id()   ==   baseLayerId)   {
                                                                baseLayer = layer; break;
```

```
                    }
                }
            }
            return baseLayer;


        function drawGroup(group) {
            var GROUP_MARGIN_BOTTOM = 45,
        GROUP_MARGIN_TOP = 10,
            GROUP_PADDING = 15,
            GROUP_BORDER = 2;

        if(typeof group == "undefined")
            return;

            var groupsDiv = $("#groups"),
        groupDiv, layersDiv, layers, key, layer;

        groupsDiv.append('<div class="group"><div
class="title-group"><h3>'+ group.name() +' ;</h3></div><div class="layers"></div></ div>');
            groupDiv = groupsDiv.last();
            layersDiv =
        groupDiv.find(".layers:last"); layers =
        group.layers();

            for(key in layers) {
                layer = layers[key];
            self.layersStore.add(layer.id(), layer);
            layersDiv.append('<div
    class="layer"><input type="checkbox" class="checkbox-layer" value="'+ layer.id()
    +'">'+ layer.rusName() +'</div>');
            }

        $("#groups").height($(window).height() -
GROUP_MARGIN_BOTTOM - GROUP_MARGIN_TOP - (2 * GROUP_PADDING) - (2
* GROUP_BORDER));
        };

        function removeMarker() {
```

```
                                        }
                                    }

                                    }

                                    function flashVersion() {
                                // Окремо визначаємо Internet Explorer var
                            ua = navigator.userAgent.toLowerCase();
                            var isIE = (ua.indexOf("msie") != -1 && ua.indexOf("opera") == - 1 &&
ua.indexOf("webtv") == - 1);
                                // Стартові змінні var
                            version = 0;
                                var newversion;
                                var lastVersion = 10; //3
                            запасом var i;
                                var plugin;

                            if (isIE) { // browser == IE try {
                                    for (i = 3; i <= lastVersion; i++) { if
                                        (eval('new
        ActiveXObject("ShockwaveFlash.ShockwaveFlash.'+i+'")')) {
                                                    version = i;
                                        }
                                    }
                                    } catch(e) {}
                                } else { // browser != IE
                                    for (i in navigator.plugins) {
                                        plugin = navigator.plugins[i];
                                        if (plugin.name == undefined) continue;
                                    if (plugin.name.indexOf('Flash') > -1) {
                                            newversion = /\d+/.exec(plugin.description);
                                        if (newversion == null) newversion = 0;
                                            if (newversion&version) version = newversion;
                                        }
                                    }
                                }
                                return version;
                            }
                        });
                        <div class="wrap">
                            <div class="content">
                                <div id="map"></div>
                                <div id="groups" class="groups">
                                </div>
                            </div> <!-- /content -->

                        </div> <!-- /wrap -->

                        </body>
                        </html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <metahttp-equiv="content-type"content="text/html;  charset=utf-8" /
>
    <title>MapEditor GS | Панель керування</title>
    <link rel="shortcut icon" type="image/png"
href="/statics/favicon/faviconRekod.png"/>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7 x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="//cdn.jsdelivr.net/npm/slick-
carousel@1.8.1/slick/slick.css"/>< /span>
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/ css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr "
crossorigin="anonymous">
    <link rel="stylesheet" href="./css/fonts.css">
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" href="./css/media.css">

    <script src="js/jquery-1.7.2.min.js" type="text/javascript"></script>
    <script src="js/jquery-ui-1.9.2.custom.min.js"
type="text/javascript"></script>
    <script src="js/jwplayer/jwplayer.js" type="text/javascript"></script>
    <script src="js/jwplayer/jwplayer.html5.js"
```

```
        type="text/javascript"></script>
                <script src="js/fancybox/jquery.fancybox.pack.js"type="text/javascript"></script>
                <script              src="js/fancybox/jquery.mousewheel-3.0.6.pack.js"
type="text/javascript"></script>
                <scriptsrc="js/fancybox/jquery.fancybox-thumbs.js"
type="text/javascript"></script>

                <script     src="http://network.geocrm.ru/public/javascripts/geoportal/geoportal-
api.min.js"type="text/javascript"></script>
                        <script src="js/app/HashMap.js" type="text/javascript"></script>
                        <script src="js/app/ModelEis.js" type="text/javascript"></script>
                        <script src="js/app/AFeatures.js" type="text/javascript"></script>
                        <script src="js/app/WMSFeatures.js" type="text/javascript"></script>
                        <script src="js/app/WFSFeatures.js" type="text/javascript"></script>
                        <script src="js/app/LeftPanel.js" type="text/javascript"></script>
                        <script src="js/app/LayerFeaturesBox.js" type="text/javascript"></script>

                        <scriptsrc="https://code.jquery.com/jquery-3.3.1.min.js"></script>

                        <script src="./js/jcore.js"></script>
                        <script src="./js/jquery.mobile.custom.min.js"></script>
                        <script src="./js/mask.js"></script>
                        <script src="./js/wow.js"></script>

                <scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-beta/
js/materialize.min.js"></script>
                        <script src="./js/main.js"></script>

                        <script>
                                $(document).ready(function(){
                                        $('.tabs').tabs();
                                });

                        </script>
                </head>
                <body>
                        <div class="admin-header">
                                <div class="admin-header-wrap">
                                        <div class="ah-left">
```

```
                                    <div class="ah-title">
                                        <a href="./admin.html">
                                            MapEditor | Адміністративна панель
                                        </a>
                                    </div>
                                </div>
                                <div class="ah-right">
                                    <div class="ah-username">
                                        <span>NVAnisimov</span>
                                    </div>
                                    <div class="ah-map">
                                        <a href="./">
                                            <img src="./img/admin-map.png" alt="">
                                            <span>Карта</span>
                                        </a>
                                    </div>
                                    <div class="ah-exit">
                                        <a href="./reg-page.html">
                                            <img src="./img/admin-exit.png" alt="">
                                            <span>Вихід</span>
                                        </a>
                                    </div>
                                </div>
                            </div>
                        </div>

                        <div class="admin-content">

                            <div class="ac-left-content">
                                <div class="acl-menu">
                                    <ul class="admin-tabs tabs">
                                        <li class="tab adm-setting">
                                            <a href="#adm-setting">
                                                <div class="icon"></div>
                                                <span class="item-
        name">Управління</spa
n>                                                  </a>
                                            <
                                            /li>

                                            <li class="tab adm-layers">
```

```
x9JvoRxT2MZw1T" crossorigin="anonymous">
        <link rel="stylesheet" type="text/css" href="//cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slick.css"/>< /span>
        <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/ css/all.css" integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr " crossorigin="anonymous">
            <link rel="stylesheet" href="./css/fonts.css">
            <link rel="stylesheet" href="./css/main.css">
            <link rel="stylesheet" href="./css/media.css">


            <scriptsrc="https://code.jquery.com/jquery-3.3.1.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0 W1" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x 0xIM+B07jRM" crossorigin="anonymous"></script>
        <script type="text/javascript" src="//cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slick.min.js"> </script>
            <script src="./js/main.js"></script>
            <scriptsrc="./js/jcore.js"></script>
            <script src="./js/jquery.mobile.custom.min.js"></script>
            <script src="./js/mask.js"></script>
            <script src="./js/wow.js"></script>
        </head>
        <body>
            <div class="main-content">
                <div class="reg-block">
                <div class="reg-title">
                    Вхід
                    </div>
                    <div class="reg-form-wrap">
                        <form action="#" class="reg-form">
                            <input type="text" placeholder="Логін" class="regInp" name="reglogin">
                            <input type="password" placeholder="Пароль"
```

```
class="regInp" name="refPass">
                                        <a href="./" class="regButton"
click="window.location='mapeditor.html'">Вхід</a>
                                        <a href="#" class="forgotPass">Забули
пароль
?</a>                                   </form>
                                </div>
                            </div>
                        </div>
                    </body>
                    </html>

                    $primary-color: #00adef;
                    $def-color: #113c4c;
                    $ green-color: #76c043;


                *::-webkit-input-placeholder        {
                    color: #aeacac;
                        opacity: 1;
                    }
                :focus::-webkit-input-placeholder          {
                    color: transparent
                    }

                :focus::-moz-placeholder        {
                    color: transparent
                    }

                :focus:-moz-placeholder        {
                    color: transparent
                    }

                :focus:-ms-input-placeholder        {
                    color: transparent
                    }

                    input::-webkit-input-placeholder{opacity:  1; transition: opacity 0.3s ease;}input::-
                moz-placeholder{opacity: 1; transition: opacity 0.3s ease;}  input:-moz-
                placeholder{opacity: 1; transition: opacity 0.3s ease;}  input:-ms-input-placeholder{opacity:
                                                1; transition: opacity 0.3s ease;}
```

```
        }
        * {
                padding: 0;
                margin: 0;
                -webkit-overflow-scrolling:
        touch; overflow-scrolling: touch;
                box-sizing: border-box;
        }
        ul { padding-left: 0px; list-style-type: none; margin: 0px; } h1,
    h2, h3, h4, h5, h6 { margin: 0px; padding: 0px; }
    a, a:hover, a:active, a:focus { color: inherit; text-decoration: none; transition:
.2s;}
        input, button { outline: none; }
        button { border: none; cursor: pointer;}
    img {max-width: 100%; }

        body {
                line-
        height: 1.3; font-
        size: 16px; color:
        #113c4c;
                font-family: "Gotham Pro", sans-
        serif; font-weight: 400;
                background-color:
        #dcdddf; letter-spacing: 0.2px;
        }

                height:
        100%;   position:
        fixed;      width:
        100%;       color:
        #4e5a61;
        }

        .content {
                position:
        relative; z-index: 1;
        }

        #map {
                height:
        100%;   position:
        fixed;      width:
        100%;
        }
        p
    re,
    code {
                border:  0
        none; font:
```

```
        inherit;
margin: 0;
        padding: 0;
        vertical-align: baseline;
}
pre code {
        background: none repeat scroll 0 0 #FFFFFF;
color: #000000;
        display: block;
}
.group {
        .title-group {
                h3 {
                        font-size:
                20px;       cursor:
                pointer;
                }
        }
}
#groups {
        width:
300px;     height:
auto;     padding:
15px;
```

```
                overflow-y:
        auto;       overflow-x:
        hidden;         z-index:
        99999;          position:
        absolute;
                background-color:
        white; right: 10px;
                top: 50px;
                border:     2px     solid
        #2E5988;  border-radius:  5px;
        opacity: 0.9;
        }
        #groups         .layers{margin:3px;}
#groups .layers .layer{padding: 5px;}
        #groups     .layers     .layer:nth-child(2n){background:#EAEFF1;}
#groups .layers .layer .checkbox-layer{ margin: 3px; }

.mapsurfer-popup-content .showFeatures { color:
        #487EAD;
                text-decoration:
        underline; cursor: pointer;
                text-align: left;
        }

.mapsurfer-popup-data
        {text-align: center;
        }

.wrap   #leftpanel-div   {
        left:  0;  position:
        fixed; top: 60px;
                z-index: 100;
        background:    #fff;
        height: 574px;
        }

.wrap #leftpanel-div #leftpanel-container { height:
        100%;
                overflow-y:
        auto; width:
```

```
.wrap #leftpanel-div #leftpanel-container .feature-group {
        padding: 10px 0px 15px 0;
                border-bottom: 1px solid
        #E1E1E1; position: relative;
        }

.wrap #leftpanel-div #leftpanel-container .feature-group h4 {
        color: #4e5a61;
                font-size:
        16px;       margin-
        bottom:       10px;
        padding-left:
        19px;
        }

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item {
        padding-left: 19px;
                paddi
        ng-top:   3px;
        position:
        relative;
        }

.wrap #leftpanel-div #leftpanel-container .feature-group .features-shown {
        border-left: 4px solid #E1E1E1;
                margin-
        bottom:       5px;
        overflow:
        hidden; padding-
        left:          15px;
        padding-right:
        2px;
        }
                margin-
        bottom:       10px;
        margin-left: 19px;
        }

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- item-
div {
                margin-left: 0px;
        }

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- item-
div-with-icon {
                padding-right: 35px;
        }

.wrap #leftpanel-div #leftpanel-container .feature-group .clicked-title { cursor:
```

```
pointer;
}

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-title {
border-bottom: 1px dashed #2C87D2;
color:
#2C87D2; margin-
bottom: 10px;
}

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content {
border-bottom: 1px solid
#E1E1E1; color: #4E5A61;
margin-
top: 10px;
padding-bottom:
5px; padding-
right: 10px;
}

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature-content
.feature-eis-photos {
height:
104px; margin-
bottom: 10px;
margin-top: 10px;
overflow: hidden;
width: 1020px;
}
.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content
.feature-eis-photos img{
border: 2px solid
#859DA9; margin-right:
1px;
}

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item p { color:
#4E5A61;
margin-
bottom: 5px;
padding-left:
14px; font-size:
12px;
}

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content
.feature-eis-files {
margin-top: 15px;
}

.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content
```

```
.feature-eis-files .item{
		background: url("../images/clip.png") no-repeat scroll 0 0 transparent; padding-left: 20px;
		margin-bottom: 5px;
	}

	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content .feature-eis-files .span-hover{
		curso
	r:	pointer;
	color:
	#487EAD;
		text-decoration: underline;
	}

	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content .feature-eis-files .span-hover:hover{
		text-decoration: none;
	}

	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content .feature-eis-files .video{
		background: url("../images/video.png") no-repeat scroll 0 0 transparent;
	}
	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content .feature-eis-files .video span{
		curso
	r:	pointer;
	color:
	#487EAD;
		text-decoration: underline;
	}
	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content .feature-eis-files .video span:hover{
		text-decoration: none;
	}

	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content .feature-eis-files .page{
		background: url("../images/page.png") no-repeat scroll 0 0 transparent;
	}

	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content p {
		padding-left: 0px;
	}
	.wrap #leftpanel-div #leftpanel-container .feature-group .feature-item .feature- content label{
		cursor: pointer;
	}
```

```
.reg-block {
        position:
absolute;    background-
color: #f8f8f8; top: 50%;
        left: 50%;
        transform:      translate(-
50%, -50%); padding: 25px;
        border: 1px #ced0d3 solid;
}
.reg-title {
        text-align: center;
        text-transform:
uppercase; font-weight:
500;
        font-size:
22px;       margin-
bottom: 25px;
}

.reg-form {
        width: 310px;
        .regInp {
                dis
        play:
        block;
        width:
        100%;
        height:
        40px;
                margin-bottom:
        10px;  background-color:
        #eae7e7; outline: none;
                border:        1px
        #dadada solid; padding:
        15px;
                trans
        ition:    .2s;
        &:nth-
        child(2) {
                        margin-bottom: 25px;
                }
                &:hover {
                        background-color: #F3F3F3;
                }

        }
        .regButton {
                dis
        play:
        block;
        width:
```

```
        100%;
            background-color:
        #fec558; color: #544e43;
            border:  1px
        #deb35f  solid;  border-
        radius:  25px;  height:
        40px;
            margin-
        bottom:  10px;
        text-align:  center;
        line-height:  40px;
        transition:  .2s;
        &:hover {
                background-color:
            #FFD88E;  border:  1px
            #FFD88E solid;
            }
        }
        .forgotPass {
            font-size: 14px;
            displa
        y: block; text-
        align:  center;
        width: 100%;
        }
}


.top-menu {
        background-
color: #fff; position:
fixed;
        top: 0;
        left: 0;
        width:
100%;
padding:  0px
5px;  z-index:
9999;
        .top-menu-list {
            font-size: 0px;
        }
        .tm-li {
            font-size:
        16px;  display:
        inline-block;
        padding:  10px
        15px;  cursor:
        pointer;  position:
        relative;
        transition:  .2s;
```

```
                &::after {
                        content
                : "; position:
                absolute;
                right: 0;
                        w
                idth:
                1px;
                height:
                50%;
                        top: 50%;
                        transform:
                translateY(-50%);
                background-color:
                #e1e1e1;
                }
                &:last-child {
                        &::after {
                                display: none;
                        }
                }
                &:hover {
                        background-color: #f1f1f1;
                }
        }
    }
    .map-block {
        height: 100vh;
    }
.table-editor-window
    {   position:
    absolute;   z-
    index: 999;
        top: 40%;
        left: 50%;
        transform:   translate(-
    50%, -50%); background-color:
    #f8f8f8; border: 1px #ced0d3
    solid; padding: 15px;
        color: #000;
    }
    .tew-work-space {
        fo
    nt-size:
    0px;
    width:
    930px;
        display:  -
    webkit-flex;
    display:    -moz-
    flex; display: -ms-
```

```
flex;  display:  -o-
flex; display: flex;
        justify-content: space-between;
}
.tew-close {
        positio
n:     absolute;
right: 15px;
        top
:      15px;
cursor:
pointer;
        border:        1px
#dedede   solid;   line-
height: 30px;
        width: 30px;
        text-align:
center;     border-
radius: 100%;
}
.tew-title {
        margin-b
```

ДОДАТОК 2

Національний технічний університет України “Київський політехнічний інститут
імені Ігоря Сікорського”
Факультет Прикладної Математики
Кафедра системного програмування і спеціалізованих комп'ютерних систем

Магістерська дисертація
на здобуття ступеня магістра за освітньо-професійною програмою

Способи підвищення ефективності засобів тестування програмних систем

Студент групи: КВ-21мп
Варган Олексій Ігорович
Керівник: Павловський В.І.

Об’єкт дослідження. Предмет дослідження. Мета роботи.

- **Об’єкт дослідження:** Процес автоматизованого тестування веб-додатків.
- **Предметом дослідження є** фреймворк для автоматизації тестування, його структура та функціональність.
- **Мета:** Розробка інноваційного фреймворку для автоматизованого тестування веб-додатків, який вирішує існуючі проблеми ефективності та масштабування.

Актуальність

- Ця тема є актуальною через зростаючу залежність сучасних бізнес-структур та організацій від веб-додатків. В контексті постійного розвитку програмного забезпечення та веб-технологій, надійність і ефективність веб-додатків стають вирішальними. Критичний аналіз існуючих рішень у сфері автоматизованого тестування вказує на певні недоліки та обмеження, що вимагають інноваційного підходу.
- Відтак, дослідження направлене на розробку вдосконаленого фреймворку для автоматизованого тестування веб-додатків є важливим для подальшого розвитку IT-сектора та пов'язаних з ним галузей.

Розробка вдосконаленого фреймворку для автоматизованого тестування веб-додатків є важливим для подальшого розвитку IT-сектора та пов'язаних з ним галузей

Для досягення цієї мети передбачено розгляд наступних завдань:

- Аналіз сучасних методів та інструментів автоматизованого тестування;
- Визначення ключових вимог до фреймворку;
- Розробка концептуальної та технічної моделі фреймворку;
- Проведення випробувань і оцінка ефективності розробленого фреймворку.

Порівняння фреймворків автоматизації тестування

Категорія/Продукт	Розроблений фреймворк	Selenium	Katalon	UFT	Test Complete	Watir
Поширеність	-	+++	++	+	++	++
Підтримка Інструменту	+++	+++	+	+	++	+
Ціна	Безкоштовний /внутрішні витрати	Безкоштовний	Безкоштовний	Дорогий	Середня ціна	Безкоштовний
Відкритість	+++	+++	-	-	-	+++
Підтримка браузерів	IE, Chrome, Firefox, Opera, Edge, мобільні браузери	IE, Chrome, Firefox, Opera, Edge, мобільні браузери	IE, Chrome, Firefox	IE, Chrome, Firefox, Safari, мобільні браузери	IE, Chrome, Firefox, Opera, Safari, Edge	IE, Chrome, Firefox, Safari, Edge
Підтримка ОС	Windows, Linux OS X	Windows, Linux OS X	Windows, Linux OS X	Windows	Windows	Windows, Linux OS X
Мова Програмування	Java	Java, C#, Perl, Python, JavaScript, Ruby, PHP	Java/Groovy	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++, C#	Ruby
CI/CD	+++	+++	++	+++	+++	+
BDD	+++	-	-	-	++	+++
Простота та зручність	+++	-	++	-	+	-

5

Основні переваги та вимоги

Підтримка інструменту.

Підтримка співробітників компанії забезпечує швидке вирішення проблем і покращення.

Умовна безкоштовність.

Витрати включають лише початкові витрати на розробку та обслуговування та витрати на розробку нових функцій. Перевага такого підходу в тому, що для розвитку використовуються внутрішні ресурси. Це означає, що будуть реалізовані лише необхідні функції і не будуть запроваджені регулярні обов'язкові платежі.

Відкритість.

Проект планує реалізувати свої ініціативу як вільне програмне забезпечення з вихідними кодами. Це надає можливість для широкого розповсюдження розробленого інструменту. Завдяки цьому, проект може отримати підтримку від сторонніх розробників, які бажають зробити свій внесок у відкритий проект. Також це може призвести й до покращення репутації оригінального розробника, завдяки участі в розвитку відкритих технологій.

6

Підтримка браузерів.

Фреймворк взаємодіє з практично всіма наявними на ринку браузерами, оскільки базується на Selenium Webdriver. Він підтримує як власні драйвери, так і ті, які розроблені сторонніми розробниками. Також є можливість створення власного драйвера для найспецифічніших і навіть пропрієтарних версій браузерів.

Мова програмування та підтримка ОС.

Для розробки фреймворку була обрана мова програмування Java, яка наразі є найбільш поширеною за низкою індексів. Наприклад, згідно із відомим індексом TIOBE Programming Community Index на травень 2018 року, мова Java посідає перше місце з рейтингом 16,38% [1]. Таким вибір дозволяє уникнути проблеми зі складнощами пошуку нових розробників для проекту. Крім того, це також впливає на потенційну кількість розробників і тестувальників, які оберуть цей фреймворк для своїх цілей.

CI/CD.

Фреймворк розробляється у формі бібліотеки, яку можна легко підключити до свого проекту шляхом вказання залежностей у настроювальних файлах систем Maven, Ant, Gradle. Це робить процес вбудовування в проект дуже простим і сприяє зручності інтеграції з середовищами CI/CD, такими як Jenkins, Hudson та ін.

BDD.

Фреймворк може здобути ще більше популярності завдяки можливості створювати автоматизовані тести за допомогою технології Behavior Driven Development (BDD). Це дозволяє фахівцям у галузі тестування програмного забезпечення, навіть тим, хто не має досвіду з жодною мовою програмування, аналізувати існуючі тести та створювати нові, використовуючи готові базові кроки.

Основні недоліки

Поширеність.

На ранніх стадіях розробки проект відомий лише вузькому колу штатних розробників і обмеженій кількості користувачів, які відкрили репозиторій проекту з відкритим кодом. Тому цей проект підтримується лише внутрішніми розробниками. Цей недолік буде виправлено, коли інструмент стане більш доступним.

Мова програмування.

Хоча Java наразі є найпоширенішою мовою, вона також може обмежити кількість користувачів, які віддають перевагу іншим популярним мовам для розробки автоматизованих тестів програмного забезпечення, таким як Python, Ruby або C#. Цей недолік частково усувається завдяки використанню інструментів технології BDD, які частково усувають потребу в спеціалізованому персоналі зі знанням мови Java.Однак повністю усунути такі потреби неможливо.

Схема процесу роботи відділу тестування «Як має бути» та її декомпозиція

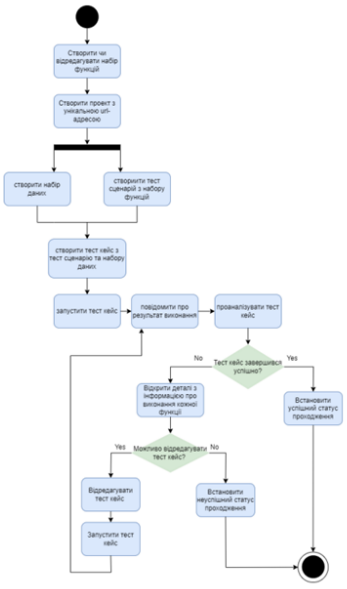
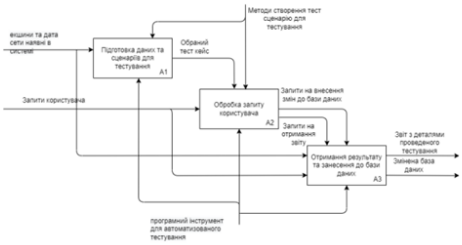


Схема процесу роботи відділу тестування «Як є» та її декомпозиція

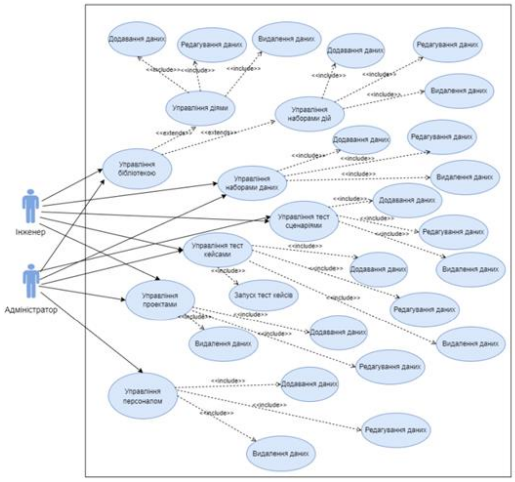


Виявлені недоліки за даною схемою процесу:

- низька продуктивність праці фахівців із тестування;
- низька якість звітної інформації через можливі помилки фахівців при ручному відображенні результатів прогонів тестів;
- висока трудомісткість тестування великих релізів;
- недосконалість організації первинних тестових даних різними співробітниками;
- недосконалість процесу аналізу проблем і помилок, що виникають при тестуванні;
- розробникам доводиться брати участь у процесі тестування на етапі смоук тестування, оскільки встановлення ПЗ, що поставляється, проводиться в ручному режимі без використання методики Continuous Integration/Delivery.

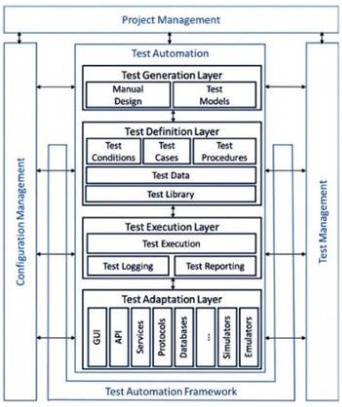
11

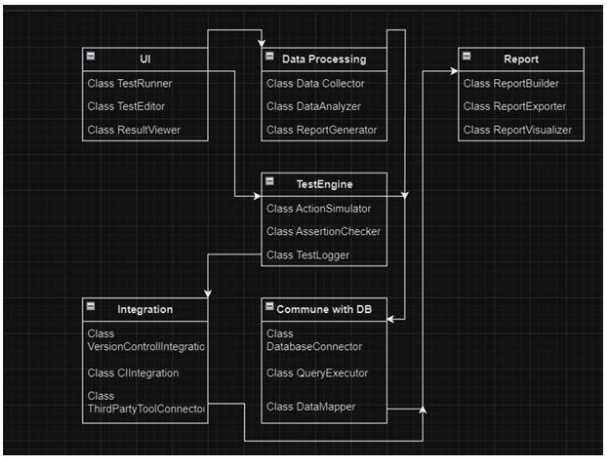
Діаграма прецедентів використання фреймворку автоматизації



12

Структура тестового фреймворка та діаграма взаємозв'язків модулів розробленого фреймворку





13

Прикладні модулі додатку

Модуль інтерфейсу користувача (UI): Цей модуль забезпечує взаємодію між користувачем та фреймворком. Він може включати інструменти для створення, редагування та запуску тестових сценаріїв, а також відображення результатів тестування. Цей модуль забезпечує інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко взаємодіяти з функціями фреймворку.

Модуль обробки даних: Відповідає за збір, обробку та аналіз даних, отриманих в результаті тестування. Цей модуль може включати алгоритми для обробки логів, збір статистики, а також забезпечення агрегації та візуалізації даних тестування.

Модуль комунікації з базами даних: Цей модуль забезпечує взаємодію фреймворку з зовнішніми базами даних, які можуть зберігати тестові сценарії, результати тестування або іншу важливу інформацію. Він може включати функції для здійснення запитів до баз даних, отримання та збереження даних.

Модуль тестового двигуна: Основа фреймворку, яка відповідає за виконання тестових сценаріїв. Включає інструменти для імітації користувацьких дій, перевірки відповідності очікуваним результатам та логування процесу тестування.

Модуль інтеграції з іншими інструментами: Цей модуль забезпечує інтеграцію з зовнішніми інструментами та сервісами, такими як системи контролю версій, інструменти неперервної інтеграції (CI/CD) та інші інструменти тестування.

Модуль звітності: Відповідає за генерацію звітів за результатами тестування. Забезпечує формування детальних звітів, які можуть включати інформацію про успішність тестів, виявлені помилки, покриття коду тестами тощо.

14

Тест кейси для перевірки працездатності функціоналу веб-додатку на прикладі онлайн магазину

- тест-кейс №1 – Перевірка проходження реєстрації,
- тест-кейс №2 – Перевірка проходження авторизації,
- тест-кейс №3 – Перевірка правильності відображення меню,
- тест-кейс №4 – Перевірка правильності відображення товарів,
- тест-кейс №5 – Замовлення товару,
- тест кейс №6 – Пошук товару,
- тест-кейс №7 – Перевірка купівлі товару,
- тест-кейс №8 – Перевірка товару на складі,
- тест-кейс №9 – Редагування ціни товару адміністратором,
- тест-кейс №10 – Перевірка сумісності із браузером Internet Explorer.

15

Результат

1. Успішно
2. Успішно
3. Успішно
4. Успішно
5. Успішно
6. Помилка. Пошук товару провалився, бо заданий товар відсутній
7. Помилка. Перевірка купівлі товару провалилась через відстуність вказаного товару
8. Помилка. Перевірка товару на складі провалилась через відсутність вказаного товару
9. Успішно
10. Помилка. Перевірка сумісності з Internet Explorer провалилась

16

Наукова новизна

Наукова новизна полягає у способі підвищення ефективності тестування шляхом інтеграції автоматизації з передовими методологіями розробки та тестування програмного забезпечення CI/CD (Continuous Integration/Continuous Delivery) та BDD (Behavior Driven Development) що дозволяє зберігти багатофункціональність та мультиплатформість й при цьому вирішує проблему поширення подібних інструментів серед фахівців, що не володіють знаннями мов(и) програмування.

17

Практична цінність

Фреймворк має значний потенціал для використання у різних сферах, де потрібне надійне тестування веб-додатків, що сприяє підвищенню їх якості та надійності. Результати дослідження можуть бути впроваджені у практику IT-компаній.

18

Апробація

- ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ XVI науково-практична конференція магістрантів та аспірантів ПМК-2023 факультету прикладної математики (28 – 30 листопада 2023 року) *Київ, Україна*. [2]

- СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ V всеукраїнська науково-практична інтернет-конференція молодих вчених та студентів (30 листопада 2023 року), *Херсон*, *Україна*. [1]

Висновок

- В даній роботі проведено аналіз фреймворків для автоматизованого тестування веб-додатків, а також технік і методологій їх тестування на різних етапах життєвого циклу розробки програмного забезпечення. В результаті було досліджено основи взаємодії клієнтської та серверної частин веб-додатків, структуру веб-сторінок, аспекти пошуку веб-елементів та методи їх тестування.
- Ключові висновки включають те, що функціональне (ручне) тестування проявляє свою ефективність при розробці нових бізнес-процесів чи проектів, у той час як автоматизоване тестування більш доцільне для перевірки існуючого функціоналу.
- Було проведено дослідження сутності автоматизованого тестування веб-додатків, і в результаті були розроблені автоматизовані сценарії тестування для онлайн магазину. Це призвело до позитивного економічного ефекту, значущого скорочення часу та витрат на ручне тестування, а також відчутного зменшення впливу людського фактора на процес тестування. Зусилля були направлені на усунення помилок з метою забезпечення якості програмного продукту.