УДК 004.415.3
DOI: 10.18523/2617-3808.2025.8.225-231

Ліпський Д. О.

АРХІТЕКТУРА НОВОЇ ВДОСКОНАЛЕНОЇ ПЛАТФОРМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ

Вдосконалення автоматизації тестування вебзастосунків є особливо актуальним напрямом у сучасному процесі розроблення програмного забезпечення. У цій статті здійснено аналіз сучасних підходів та інструментів автоматизації тестування, їхніх переваг та недоліків. Розглянуто шляхи усунення цих недоліків, а також можливість інтеграції технологій штучного інтелекту.

У статті представлено архітектуру запропонованої платформи автоматизованого тестування, реалізованої у формі бібліотеки, що легко інтегрується в наявні проєкти. Архітектуру побудовано за модульним принципом, що забезпечує гнучкість, масштабованість і можливість поетапного розширення функціональності. Основні компоненти платформи — конфігураційний модуль, менеджер драйверів, модулі взаємодії з елементами, логування, API-викликів та роботи з локальним сховищем браузера — працюють як єдиний узгоджений механізм, забезпечуючи прозоре, стабільне та ефективне виконання тестів.

Окрему увагу приділено аналізу таких аспектів сучасних технологій автоматизації, як зниження вартості впровадження та підтримки тестових рішень, масштабованість, гнучкість налаштувань, а також інтеграція з іншими компонентами життєвого циклу програмного забезпечення.

Ключові слова: автоматизація, вебтехнології, платформа, тестування.

Вступ

Завдання тестування вебзастосунків полягає у створенні та виконанні тестових сценаріїв на вебплатформах за допомогою спеціалізованого програмного забезпечення (ПЗ). Така технологія перевірки правильності ПЗ спрямована на виявлення помилок і невідповідностей до вимог у вебзастосунках [1]. Автоматизація дозволяє виконувати повторювані тестові процедури без залучення тестувальника, що значно підвищує ефективність перевірки якості та надійності програмних продуктів [2].

Основні компоненти автоматизованого тестування охоплюють: визначення тестових даних і наперед заданих умов, встановлення очікуваних результатів і безпосереднього виконання тестових сценаріїв. Це забезпечує виявлення та вирішення таких проблем у вебзастосунках, як функціональне тестування, перевірка навантаження, безпеки, користувацького інтерфейсу та інтеграції систем [5].

Під час функціонального тестування перевіряють вибрані функції вебзастосунку через імітацію реальних сценаріїв користувача, включаючи тестування форм, аутентифікації та пошукових запитів [1]. Також здійснюється навантажувальне тестування для оцінки стійкості застосунку під великою кількістю користувачів, використовуючи віртуальне середовище для генерації численних запитів до системи [5].

Безпека є однією з критичних областей, де тестування зосереджене на виявленні потенційних уразливостей, таких як SQL ін’єкції і крос-сайтовий скриптинг [4]. Це забезпечує захист вебзастосунку від шкідливих дій. Тестування користувацького інтерфейсу перевіряє, що візуальні елементи добре інтегровані та інтуїтивно зрозумілі для користувачів, тоді як інтеграційне тестування забезпечує, що всі модулі системи ефективно працюють разом [5].

Серед сучасних інструментів для автоматизації тестування, таких як Selenium, QTP і Ranorex, розробники можуть вибрати ті, що підтримують широкий спектр мов програмування та можливість інтеграції з різними середовищами розробки [2]. Ці інструменти забезпечують гнучкість і адаптивність, необхідні для адекватного реагування на специфічні вимоги проєкту. Однак наявні методи і технології автоматизації тестування лише частково покращують якість вебзастосунків і знижують можливі ризики. Тому актуальними є розроблення і впровадження нових, більш ефективних рішень на основі AI [5].

Переваги та недоліки сучасних платформ автоматизованого тестування

Автоматизація тестування забезпечує численні переваги, які значно покращують процес розроблення програмного забезпечення: збільшення швидкості виконання тестів, підвищення точності, економія ресурсів та покращення покриття тестування [1].

Одна з найважливіших переваг автоматизації тестування — здатність швидко проводити великі обсяги тестів, виконуючи тестові сценарії значно оперативніше, ніж це могли б зробити люди. Це призводить до прискорення циклу розробки, оскільки зворотний зв'язок отримується майже відразу, що дає змогу швидше вносити корективи [5].

Автоматизація вивільняє цінні ресурси, зокрема час тестувальників, які тепер можуть бути спрямовані на більш складні стратегічні завдання. Замість витрат часу на рутинне виконання тестів тестувальники можуть зосередитися на аналізі складних випадків, вдосконаленні тестових сценаріїв та оптимізації загальної стратегії якості [5].

Автоматизація мінімізує помилки тестувальників, що часто трапляються при ручному виконанні через втому або неуважність. Кожен автоматизований тест виконується однаково при кожному запуску, забезпечуючи стабільність та повторюваність результатів, що гарантує виявлення та усунення помилок до того, як продукт досягне кінцевого користувача [2].

Оскільки автоматизація потребує меншого втручання людей, витрати на трудові ресурси зменшуються, роблячи процес тестування більш економічно вигідним. Автоматизація також дає можливість працювати неперервно, що збільшує загальний обсяг виконаних робіт за певний період [5].

Автоматизація дає змогу включати у тестування значно більшу кількість тестових сценаріїв, ніж у ручному режимі, що забезпечує ширше покриття і детальнішу перевірку програмного забезпечення [5]. Використання штучного інтелекту (AI) в платформах для автоматизації тестування вебзастосунків є ключовим напрямком підвищення якості, оскільки AI та машинне навчання (ML) дають можливість розширити можливості традиційного автоматизованого тестування, забезпечуючи більшу точність, ефективність і гнучкість процесів тестування [5].

Інтеграція AI в автоматизацію тестування сприяє суттєвому покращенню різних аспектів цього процесу. По-перше, AI підвищує ефективність тестування за рахунок автоматичного аналізу результатів тестів, допомагаючи ідентифікувати закономірності та передбачати потенційні проблеми, зменшуючи потребу в ручному аналізі [1]. Важливим є також оптимізований вибір тестових сценаріїв: використовуючи методи ML, платформа може визначати, які сценарії найкраще підходять для конкретних змін у коді, що робить тестування більш ефективним і цілеспрямованим [4].

Крім того, штучний інтелект сприяє автоматичному виявленню та класифікації дефектів. Це допомагає швидко знаходити помилки у програмному забезпеченні та оцінювати їх серйозність і тип, що значно спрощує процес виправлення [5].

Нарешті, існуючі AI рішення аналізують історичні дані тестування, виявляючи слабкі місця в тестових сценаріях і пропонуючи можливі покращення. Це підвищує загальну якість тестових процедур і сприяє створенню більш надійного програмного забезпечення [2].

Загалом, автоматизація тестування пропонує значні переваги, які забезпечують вищу продуктивність та ефективність процесів розроблення програмного забезпечення, зменшуючи при цьому витрати та забезпечуючи високу якість продуктів [1].

Автоматизація тестування потребує значних початкових інвестицій, що передбачають витрати на придбання та налаштування спеціалізованого програмного забезпечення, необхідного для реалізації автоматизованих процесів [1]. Наприклад, платформи на зразок BrowserStack можуть бути особливо дорогими для невеликих команд або індивідуальних розробників, що створює додаткове фінансове навантаження [5].

Також для створення систем тестування потрібні відповідні інструменти, що підтримуватимуть автоматизацію на всіх етапах. Важливим аспектом є розвиток компетенцій: необхідно підвищити кваліфікацію наявного персоналу або залучити нових фахівців із потрібними знаннями та навичками [4]. Такі спеціалісти повинні мати досвід розробки й підтримки автоматизованих систем, що забезпечить стабільне функціонування та розвиток автоматизації тестування в майбутньому [5].

Автоматизовані тестові системи вимагають регулярного оновлення, щоб відповідати змінам у програмному забезпеченні, яке вони тестують. Завдання такого оновлення може бути складним і ресурсоємним. Наприклад, платформи, що часто оновлюють свій UI, як-от Browserling, вимагають постійного оновлення тестових скриптів для взаємодії з новими елементами інтерфейсу [1].

Автоматизація тестування може бути особливо складною при імітації людської взаємодій, часто необхідної у комплексних тестових сценаріях. Такі інструменти, як Appium або Robot Framework, пропонують рішення для деяких сценаріїв, але вони не відтворюють складні аспекти користувацького досвіду, що обмежує покриття тестів [5].

Зміни в користувацькому інтерфейсі вебзастосунків відбуваються часто, що своєю чергою вимагає реалізації можливості постійного оновлення тестових скриптів, особливо залежних від специфічних атрибутів елементів, таких як ідентифікатори, класи, або стилі. Ці оновлення можуть бути частими та потребувати значних зусиль для забезпечення того, щоб тестові скрипти залишалися актуальними та ефективними [4].

Ці аспекти підкреслюють, що автоматизація тестування хоча й є могутнім інструментом для покращення ефективності та об'єму тестування, та має свої значні виклики і недоліки, які вимагають уважного розгляду та планування перед розробленням і впровадженням.

Інтеграція штучного інтелекту (AI) в платформи для автоматизації тестування вебзастосунків стає ключовим напрямом розвитку в галузі якості програмного забезпечення. Один із прикладів цього — партнерство Microsoft та Leapwork, що спрямоване на надання можливостей для автоматизації тестування користувачам Microsoft Dynamics 365 і Microsoft Power Platform. Вони використовують платформу Leapwork, яка є AI-підсиленою, візуальною та забезпечує безпеку, для спрощення процесу створення тестів і зниження ризиків збоїв під час щомісячних оновлень програмного забезпечення [2]. Ще один приклад — компанія Tricentis, яка використовує технологію швидкого оптичного розпізнавання символів (OCR) для автоматизації тестування, це покращує точність та швидкість процесу за допомогою візуального аналізу [5].

Архітектура та особливості нової платформи для автоматизованого тестування вебзастосунків

На основі проведеного в попередньому розділі аналізу сильних і слабких сторін сучасних платформ запропоновано нову архітектуру автоматизованої тестової системи, реалізованої у вигляді бібліотеки, як спробу подолати ключові обмеження, що виникають під час використання типових підходів. У розробленій платформі зроблено акцент на модульність, простоту конфігурації, підтримку паралельного тестування, централізоване керування параметрами середовища та гнучку інтеграцію з різними рівнями системи, зокрема через API.

Система побудована з використанням мови C# та принципів об'єктно-орієнтованого програмування. Основу архітектури становить WebDriverManager, що реалізує патерн Singleton для уникнення надмірного створення браузерних сесій, зберігаючи стабільність тестів і оптимізуючи споживання ресурсів. Підтримка роботи як у локальному, так і у віддаленому середовищі (через Selenium Grid) забезпечує масштабованість платформи. Конфігурація параметрів тестування здійснюється через зовнішні JSON-файли, що дає можливість змінювати налаштування без втручання у код, що є особливо актуальним у багатокомандних або мультисередовищних проєктах.

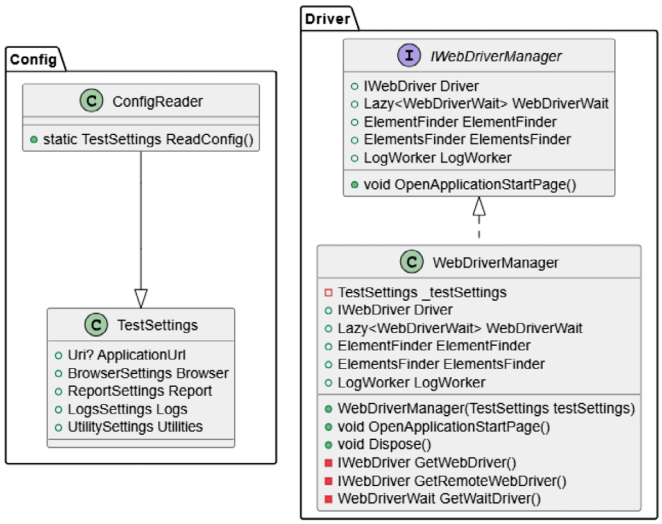

Рис. 1. Конфігурація і драйвери

Платформа підтримує як UI-, так і API-тестування, що дозволяє реалізовувати комплексні перевірки багаторівневих вебзастосунків. Для роботи з вебелементами створено набір класів (Element, ElementFinder, ElementsFinder), які розширюють функціональність Selenium WebDriver завдяки вбудованим механізмам автоматичного скролінгу, підсвічуванням елементів, розширеним перевіркам атрибутів і динамічним очікуванням (через інтеграцію з власним модулем Wait). Це суттєво підвищує стабільність автоматизованих тестів, особливо у випадках із динамічним інтерфейсом, що активно використовує AJAX-запити або асинхронне динамічне відображення контексту на вебсторінках.

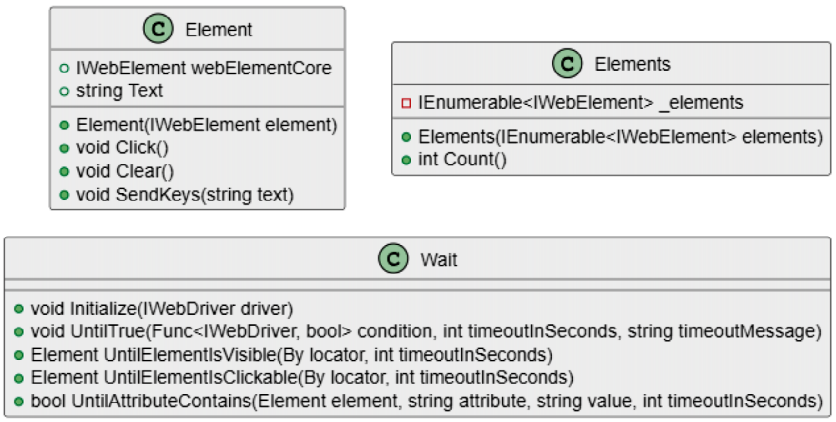

Рис. 2. Робота з вебелементами

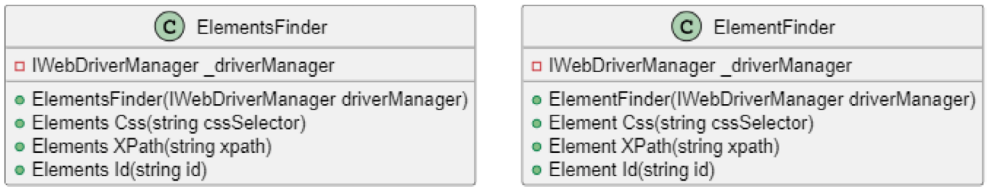

Рис. 3. Пошук вебелементів

Модуль ApiWorker забезпечує повноцінну підтримку API-тестування, дозволяючи надсилати HTTP-запити, обробляти й перевіряти відповіді. Платформа орієнтована на асинхронне виконання запитів, що дозволяє виконувати паралельні перевірки та значно скорочує час проходження тестів. Додатково реалізовано LocalStorageWorker для роботи з локальним сховищем браузера — важливого для тестування додатків, які зберігають користувацький стан безпосередньо в клієнті.

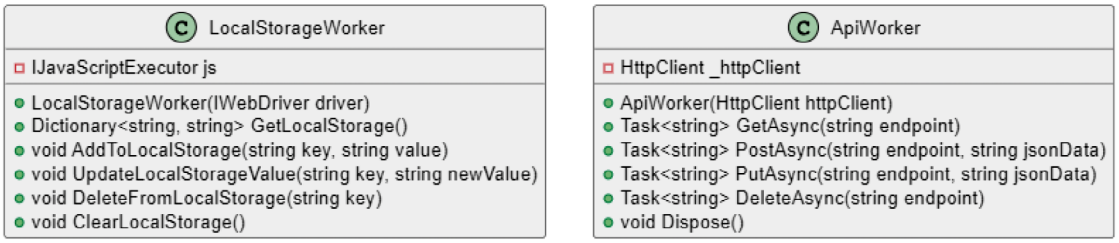

Рис. 4. Допоміжні модулі платформи — частина 1

Для підвищення гнучкості й варіативності сценаріїв застосовується TestDataWorker, який генерує унікальні дані (рядки, числа, дати, email, телефонні номери тощо), що дозволяє тестувати поведінку системи в умовах реалістичних змінних. Результати тестування перевіряються за допомогою модулів CheckWorker та VerifyWorker, які підтримують як базові, так і складні перевірки (наприклад, порівняння списків незалежно від порядку, перевірка регулярних виразів або повторне виконання перевірок через задані проміжки часу).

Для фіксації перебігу тестування та спрощення аналізу результатів реалізовано LogWorker, який записує структуровані лог-файли з часовими мітками, типами повідомлень і джерелом виклику. Це дозволяє легко ідентифікувати джерело помилки та надає прозорість під час перевірки якості програмного забезпечення.

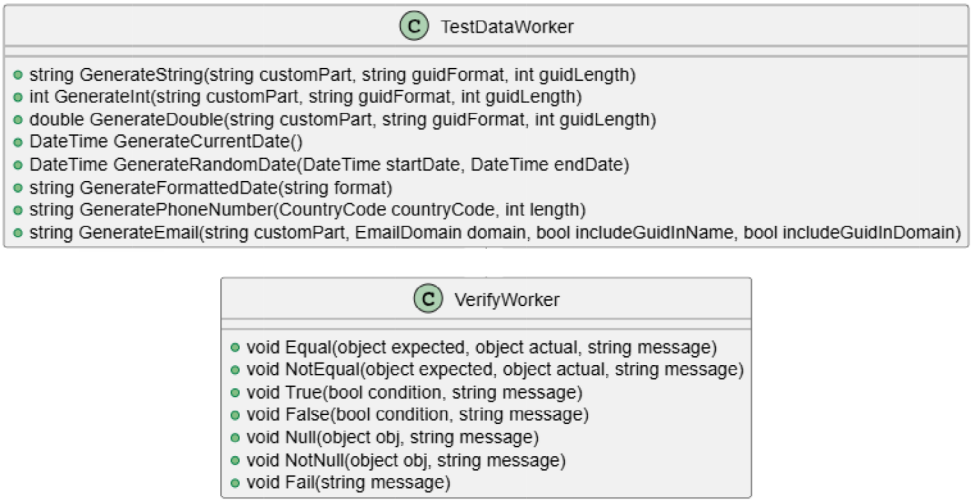

Рис. 5. Допоміжні модулі платформи — частина 2

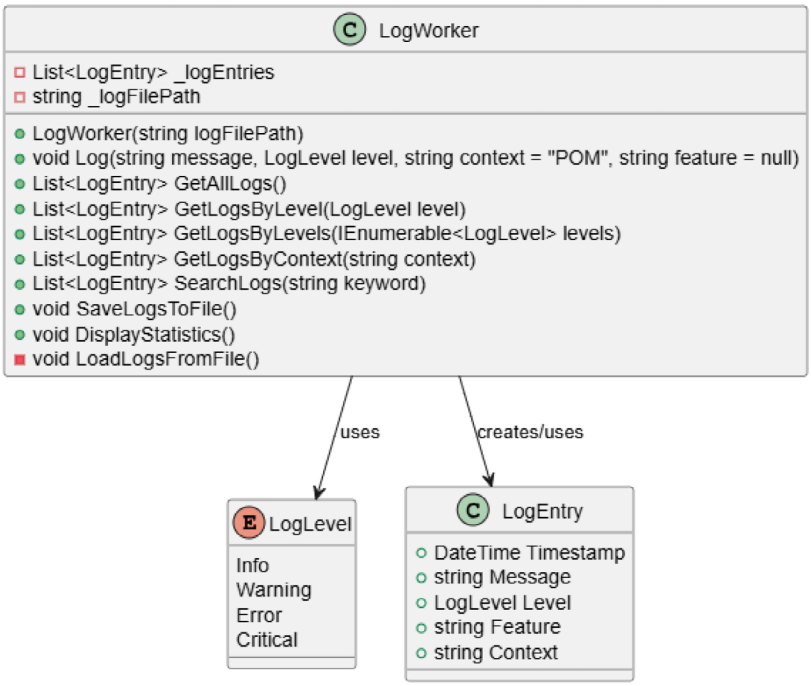

Рис. 6. Логування подій

Для візуалізації результатів використовується HtmlReportGenerator, який генерує інтерактивні HTML-звіти з маркуванням кроків і деталями кожного сценарію.

Однією з важливих переваг архітектури запропонованої платформи є її відкритість до інтеграції інтелектуальних рішень. Розвиток системи передбачає поетапне впровадження сучасних моделей машинного навчання, зокрема логістичної регресії, методів кластеризації, дерев рішень, випадкових лісів, штучних нейронних мереж та машин опорних векторів (SVM), які вже зарекомендували себе як ефективні інструменти для задач класифікації, виявлення аномалій, побудови моделей ризику та аналізу структурованих логів [6].

У рамках реалізації цієї стратегії вже впроваджено окремий модуль для оброблення логів, який забезпечує можливість інтеграції алгоритмів машинного навчання з метою автоматизованої класифікації дефектів, визначення їх пріоритетності та прогнозування потенційних збоїв. Завдяки модульності архітектури, функціональність системи може розширюватися без внесення змін до її основної логіки. Оскільки зазначений напрям охоплює окрему дослідницьку область, детальний опис використаних підходів, алгоритмів та результатів їхнього застосування буде представлено в подальших публікаціях.

Загальна ефективність платформи забезпечується не лише підтримкою інтелектуальних рішень, а й продуманою архітектурною будовою її основних модулів, кожен з яких виконує чітко визначену функцію та має низку технічних переваг.

Перевагою ініціалізаційного блоку є його гнучкість і конфігурованість. Компонент ConfigReader зчитує параметри тестового середовища та передає їх у TestSettings, що дозволяє швидко адаптувати платформу до різних середовищ і сценаріїв без потреби змінювати код. Це значно підвищує масштабованість та повторне використання тестів.

Модуль WebDriverManager забезпечує централізоване створення екземпляра браузера та відкриття стартової сторінки. Його головна перевага — уніфікований механізм роботи з різними браузерами, що підвищує стабільність тестових запусків і спрощує організацію крос-браузерного тестування.

Сильним боком архітектури є чіткий поділ відповідальностей. Окремі модулі для пошуку та взаємодії з елементами забезпечують підвищену надійність та простоту супроводу тестів. Завдяки цьому зменшується ризик помилок під час виконання та спрощується оновлення тестів у разі змін в інтерфейсі.

Ще однією перевагою платформи є інтеграція перевірок, логування та очікувань у режимі реального часу. Це забезпечує високу прозорість процесу тестування, полегшує аналіз результатів і дає змогу оперативно виявляти проблемні місця.

Модуль ApiWorker підвищує функціональну повноту платформи, дозволяючи реалізовувати інтеграційні сценарії через REST-запити. Це розширює сферу застосування платформи та дозволяє поєднувати UI- та API-рівні перевірок у межах одного тесту.

Компонент LocalStorageWorker забезпечує роботу з локальним сховищем браузера, що є особливо корисним для тестування сценаріїв зі збереженням стану. Його інтеграція дозволяє реалізовувати перевірки, які імітують поведінку реального користувача, що є додатковою перевагою при тестуванні складних вебзастосунків.

Основні модулі розробленої платформи пройшли успішну апробацію при виконанні лабораторних робіт з автоматизованого тестування вебзастосунків студентами Київського національного університету імені Тараса Шевченка.

Створена цілісна інтеграція модулів є надійним середовищем для автоматизованого тестування, яке легко адаптується до вимог швидкого впровадження, підтримки та масштабування. Таким чином, описані в роботі результати демонструють переваги запропонованої архітектури як універсального рішення для створення гнучких, розширюваних і ефективних систем автоматизованого тестування вебзастосунків, придатних для практичного використання в освітньому та промисловому середовищі.

Висновок

Створення нової платформи автоматизованого тестування зумовлене необхідністю усунення наявних недоліків, які заважають задовольнити вимоги, що стрімко зростають, до розроблення систем програмного забезпечення.

Реалізована платформа мінімізує початкові витрати, пов'язані з автоматизацією, та надає комплексне рішення «з коробки», усувається потреба вкладати кошти в розроблення власних інструментів для тестування. Дизайн платформи зосереджений на масштабованості, що дозволяє легко розширювати набори тестів або функціонал без значних додаткових інвестицій.

Інтеграція з інструментами типу Specflow підсилює адаптивність платформи, даючи змогу користувачам мінімальними зусиллями оновлювати тестові сценарії у відповідь на розвиток функціоналу додатків або вимог. Такий підхід не лише зменшує навантаження на підтримку, а й гарантує, що тести залишаються актуальними та ефективними у довгостроковій перспективі.

Модулі розроблюваної платформи долають загальні технічні обмеження, інтегруючи API та прямі запити до баз даних, що полегшує тестування складних сценаріїв, які можуть бути важкими для традиційної взаємодії через UI. Такий стратегічний вибір розширює обсяг тестування, гарантуючи всебічне покриття, яке охоплює процеси на рівні бекенду, цілісність даних і більше, незалежно від користувацького інтерфейсу.

Подальший розвиток платформи дасть можливість інтегрувати передові AI та ML моделі, такі як логістична регресія, методи кластеризації, дерева рішень, випадкові ліси, нейронні мережі та машини опорних векторів (SVM). Ці моделі можуть бути адаптовані для покращення процесів тестування шляхом передбачення потенційних збоїв, оптимізації вибору тестових кейсів та автоматичної ідентифікації дефектів.

Список літератури

1. Forgacs I. Modern Software Testing Techniques: A Practical Guide for Developers and Testers / I. Forgacs, A. Kovacs. — 1st ed. — Apress, 2024. — 350 p.
2. Homes B. Fundamentals of Software Testing / B. Homes. — 2nd ed., Revised and Updated. — Wiley-ISTE, 2024. — 280 p.
3. Lipskyi D. O. Development of a Platform for Web Application Testing Automation / D. O. Lipskyi // Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics. — 2023. — No. 1. — Pp. 45–50.
4. Loubser N. Software Engineering for Absolute Beginners: Your Guide to Creating Software Products / N. Loubser. — 1st ed. — Apress, 2021. — 330 p.
5. Mohan G. Full Stack Testing: A Practical Guide for Delivering High Quality Software / G. Mohan. — O'Reilly Media, 2022. — 420 p.
6. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / A. Géron. — 3rd ed. — O'Reilly Media, 2022. — 856 p.

References

Forgacs, I., & Kovacs, A. (2024). *Modern software testing techniques: A practical guide for developers and testers* (1st ed.). Apress.

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (3rd ed.). O'Reilly Media.

Homes, B. (2024). *Fundamentals of software testing* (2nd ed., Revised and updated). Wiley-ISTE.

Lipskyi, D. O. (2023). Development of a platform for web application testing automation. *Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics*, *1*, 45–50.

Loubser, N. (2021). *Software engineering for absolute beginners: Your guide to creating software products* (1st ed.). Apress.

Mohan, G. (2022). *Full stack testing: A practical guide for delivering high quality software*. O'Reilly Media.

D. Lipskyi

ARCHITECTURE OF A NEW ENHANCED PLATFORM FOR AUTOMATED WEB APPLICATION TESTING

Improving the automation of web application testing is a particularly relevant and rapidly evolving area in the modern software development process. The growing complexity of web-based systems, the increasing frequency of release cycles, and the ever-rising demand for high software reliability and performance make the adoption of automated testing solutions not only desirable but essential. This paper analyses the current state of test automation technologies, with a focus on widely adopted tools, frameworks, and methodologies. It outlines their primary advantages, including enhanced speed, high repeatability, improved accuracy, and reduced human error. At the same time, it identifies common limitations, such as high initial setup and learning costs, challenges in test maintenance, and limited adaptability to rapidly changing or project-specific requirements.

To address these challenges, the article introduces a novel architecture for an automated testing platform, designed and implemented as a reusable, extensible software library. The platform is built on a modular architecture, ensuring flexibility, maintainability, scalability, and seamless integration into a wide range of existing web development projects. Its core modules include a configuration handler, a browser driver manager, components for interaction with UI elements, a centralised logging subsystem, API communication tools, and browser storage management capabilities. These modules function together as a cohesive unit, forming a reliable and transparent environment that facilitates efficient and robust test execution.

In addition to architectural innovations, the paper discusses strategies to enhance test maintainability and reduce long-term resource consumption. These include intelligent reuse of test components, support for parameterised configurations, and mechanisms for simplifying test orchestration and execution across different environments. The proposed solution provides a practical, scalable framework for improving the quality, reliability, and efficiency of web application testing.

Keywords: automation, web technologies, platform, testing.

Матеріал надійшов 08.06.2025