Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)

Кафедра Інформаційно-вимірювальних технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів тестування вебдодатків
(тема)

Виконав:
здобувач 2 року навчання,
групи ЗЯм-24-1
Данило Лариков
(власне ім'я, прізвище)

Спеціальність 175 – Інформаційно-вимірювальні технології
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Забезпечення якості
(повна назва освітньої програми)

Керівник проф. Андрій ЄГОРОВ
(посада, власне ім'я, прізвище)

Допускається до захисту
Завідувач кафедри _____ Ігор ЗАХАРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій

Кафедра Інформаційно-вимірювальних технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 175 – Інформаційно-вимірювальні технології
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Забепечення якості
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ларикову Данилу Сергійовичу
(прізвище, ім'я, по батькові)

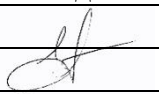
1. Тема роботи Дослідження методів тестування вебдодатків
затверджена наказом по університету від «07» листопада 2025 р. № 1011Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії «15» грудня 2025 р.
3. Вихідні дані до роботи Для виконання роботи необхідно провести теоретичний огляд методів тестування веб-додатків та здійснити практичне дослідження їх застосування на прикладі реального веб-додатку e-commerce. У практичній частині потрібно виконати функціональне тестування, юзабіліті-тестування, оцінювання продуктивності, тестування кросбраузерної сумісності та базові перевірки безпеки. Для аналізу застосовуються DevTools, Google Lighthouse, локальні браузери, а також сервіси для документування тестування (TestCaseLab тощо).
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ та загальна характеристика тестування веб-додатків: роль тестування у життєвому циклі програмного забезпечення та його значення для забезпечення якості. 4.2 Класифікація веб-додатків та методів їх тестування: функціональні та нефункціональні види тестування, особливості їх застосування для сучасних веб-систем. 4.3 Характеристики якості веб-додатків та параметри їх оцінювання відповідно до стандарту ISO/IEC 25010. 4.4 Особливості тестування веб-додатків: клієнт-серверна архітектура, браузерна залежність, динамічний контент, вимоги до продуктивності та безпеки. 4.5 Характеристика об'єкта дослідження: опис веб-додатку Rozetka.ua, основні функціональні можливості та користувацькі сценарії. 4.6 Обґрунтування вибору методів і видів тестування для оцінювання якості веб-додатку. 4.7 Проведення функціонального

тестування веб-додатку: розробка та використання чек-листа і тест-кейсів для перевірки ключових бізнес-функцій. 4.8 Проведення нефункціонального тестування веб-додатку: оцінювання юзабіліті, продуктивності, кросбраузерної сумісності та базових аспектів безпеки. 4.9 Аналіз результатів тестування та оцінка якості роботи веб-додатку за отриманими показниками. 4.10 Формування висновків щодо ефективності застосованих методів тестування веб-додатків. 4.11 Загальні висновки за результатами виконання кваліфікаційної роботи. 4.12 Перелік джерел посилання.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

5.1 Поняття та призначення тестування програмного забезпечення. 5.2 Класифікація методів тестування програмного забезпечення. 5.3 Функціональне та нефункціональне тестування. 5.4 Веб-додатки як об'єкт тестування. 5.5 Основні параметри якості веб-додатків. 5.6 Основні параметри якості веб-додатків.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основний	Проф. Єгоров А. Б.		10.01.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1.	Отримання завдання на кваліфікаційну роботу	10.11.2025	Виконано
2.	Аналіз наукових і навчальних джерел з тестування веб-додатків, вивчення методів та класифікацій	11.11.2025 –16.11.2025	Виконано
3.	Визначення особливостей веб-додатків як об'єктів тестування та аналіз характеристик якості	17.11.2025	Виконано
4.	Вибір об'єкта дослідження та аналіз функціональних можливостей веб-додатку Rozetka.ua	18.11.2025 – 21.11.2025	Виконано
5.	Обґрунтування вибору методів та видів тестування для практичної частини роботи	22.11.2025	Виконано
6.	Проведення функціонального тестування веб-додатку	23.11.2025 – 27.11.2025	Виконано
7.	Проведення нефункціонального тестування	28.11.2025 – 01.12.2025	Виконано
8.	Аналіз результатів тестування та узагальнення отриманих даних	02.12.2025 – 03.12.2025	Виконано
9.	Оформлення пояснювальної записки відповідно до вимог нормоконтролю	04.12.2025 – 07.12.2025	Виконано
10.	Підготовка графічного матеріалу та презентації для захисту роботи	08.12.2025 – 09.12.2025	Виконано

11.	Подання роботи на перевірку	11.12.2025	Виконано

Дата видачі завдання «10» листопада 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Андрій ЄГОРОВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи магістра: n с., 26 рис., 5 табл., n джерел.

МЕТОДИ ТЕСТУВАННЯ, ВЕБ-ДОДАТКИ, ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ, НЕФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ, ПРОДУКТИВНІСТЬ, ЮЗАБІЛІТІ, КРОСБРАУЗЕРНІСТЬ, БЕЗПЕКА, E-COMMERCE, ЗАБЕЗПЕЧЕННЯ ЯКОСТІ.

Об’єктом дослідження кваліфікаційної роботи є процес тестування веб-додатків у контексті сучасних e-commerce систем.

Предметом дослідження є методи та підходи до оцінювання якості веб-додатків, що охоплюють функціональні й нефункціональні аспекти їх роботи.

Метою дослідження є аналіз методів тестування веб-додатків та їх практичне застосування для оцінки якості реального веб-додатку Rozetka.ua.

Область застосування – процеси забезпечення якості веб-систем, зокрема високонавантажених e-commerce платформ.

Результати роботи – систематизовано класифікацію веб-додатків та методів їх тестування; виконано аналіз характеристик, що впливають на побудову тестування. На практичному прикладі Rozetka.ua проведено функціональні перевірки (чек-лист, тест-кейси), досліджено юзабіліті, продуктивність, кросбраузерну сумісність та базові аспекти безпеки. Отримані результати підтвердили коректність роботи ключових модулів веб-додатку, стабільність продуктивності та відповідність інтерфейсу сучасним UX-вимогам, а також показали важливість комплексного підходу до тестування динамічних веб-систем.

ABSTRACT

Explanatory note of the Master's qualification work: n pages, 26 figures, 5 tables, n references.

TESTING METHODS, WEB APPLICATIONS, FUNCTIONAL TESTING, NON-FUNCTIONAL TESTING, PERFORMANCE, USABILITY, CROSS-BROWSER COMPATIBILITY, E-COMMERCE, QUALITY ASSURANCE.

The object of the research is the process of testing web applications in the context of modern e-commerce systems.

The subject of the research is the methods and approaches for assessing the quality of web applications, including both functional and non-functional aspects of their operation.

The scope of application covers quality assurance processes for web systems, particularly large-scale e-commerce platforms. The results of the work include a systematization of web application classifications and testing methods, as well as an analysis of characteristics affecting the design of testing processes. A practical study was conducted on Rozetka.ua, involving functional testing (checklist and test cases), usability evaluation, performance assessment, cross-browser compatibility testing, and basic security checks. The obtained results confirmed the correct operation of key modules, stable performance indicators, and compliance of the interface with modern UX standards. The findings highlight the importance of a comprehensive testing approach for dynamic web systems.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – Програмне забезпечення

API – Програмний інтерфейс додатка (Application Programming Interface)

HTTP/HTTPS – Протокол передавання гіпертексту / Захищений протокол передавання (HyperText Transfer Protocol / Secure)

JS – Мова JavaScript

SQL – Структурована мова запитів (Structured Query Language)

QA – Забезпечення якості (Quality Assurance)

QC – Контроль якості (Quality Control)

SDLC – Життєвий цикл розробки ПЗ (Software Development Life Cycle)

STLC – Життєвий цикл тестування ПЗ (Software Testing Life Cycle)

ISTQB – Міжнародна рада з кваліфікації тестування програмного забезпечення (International Software Testing Qualifications Board)

OWASP – Проєкт веб-безпеки з відкритим кодом (Open Web Application Security Project)

SPA – Односторінковий веб-додаток (Single Page Application)

PWA – Прогресивний веб-додаток (Progressive Web Application)

UI – Інтерфейс користувача (User Interface)

UX – Користувацький досвід (User Experience)

FCP – Перше відображення контенту (First Contentful Paint)

LCP – Найбільший елемент контенту (Largest Contentful Paint)

CLS – Кумулятивне зміщення макета (Cumulative Layout Shift)

TBT – Загальний час блокування (Total Blocking Time)

ВСТУП

У сучасному цифровому середовищі веб-додатки стали фундаментальною частиною майже всіх сфер діяльності – електронної комерції, освіти, фінансових сервісів, державних послуг та мультимедійних платформ. Зростання обсягу онлайн-транзакцій, підвищення вимог до зручності використання та збільшення конкуренції між сервісами роблять питання забезпечення якості веб-додатків надзвичайно актуальним.

Стабільність роботи веб-додатку, його швидкодія, безпека, інтуїтивність інтерфейсу та коректність бізнес-логіки безпосередньо впливають на довіру користувачів і комерційні показники компаній. Саме тому тестування є одним з ключових етапів життєвого циклу програмного забезпечення, а його методи постійно вдосконалюються відповідно до розвитку технологій та архітектурних рішень.

Особливу увагу привертають високонавантажені e-commerce системи, які обробляють десятки тисяч запитів одночасно, підтримують динамічні інтерфейси, інтерактивні каталоги, особисті кабінети користувачів та складні бізнес-процеси замовлення товарів. Однією з таких систем є веб-додаток Rozetka.ua – найбільший український онлайн-ретейлер, функціонування якого вимагає високої надійності та якості реалізації інтерфейсних і серверних компонентів.

У межах даної кваліфікаційної роботи розглядаються методи функціонального та нефункціонального тестування веб-додатків, їх класифікація, особливості застосування та роль у підвищенні якості сучасних веб-систем. Особливу увагу приділено оцінюванню продуктивності, юзабіліті, кросбраузерної сумісності та базових показників безпеки, що визначають загальний користувацький досвід і стабільність роботи веб-додатку.

Практична частина роботи присвячена аналізу та тестуванню веб-додатку Rozetka.ua, включаючи проведення функціональних перевірок,

дослідження продуктивності за допомогою інструмента Lighthouse, оцінку UX-характеристик, аналіз кросбраузерної поведінки й перевірку клієнтської безпеки.

Метою дослідження є аналіз і практичне застосування сучасних методів тестування веб-додатків для комплексного оцінювання якості реальної e-commerce системи.

1 ТЕОРЕТИЧНІ ОСНОВИ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Поняття та призначення тестування програмного забезпечення

Тестування програмного забезпечення – це процес перевірки функціонування програмного продукту з метою виявлення дефектів, оцінювання його якості та підтвердження відповідності встановленим вимогам. Основною метою тестування є гарантування того, що система працює коректно, стабільно та очікувано в реальних умовах використання [2].

У контексті забезпечення якості тестування виконує три ключові функції:

- виявлення дефектів, що можуть вплинути на працездатність або надійність системи;
- підтвердження відповідності розробленого функціоналу технічним та бізнес-вимогам;
- зменшення ризиків помилок на етапі експлуатації та підвищення задоволеності користувачів.

У процесі забезпечення якості програмного забезпечення розрізняють три взаємопов’язані дисципліни: Quality Assurance (QA), Quality Control (QC) та Testing. Попри те, що ці поняття часто плутають, вони виконують різні функції в межах життєвого циклу розроблення ПЗ. Їхнє співвідношення можна подати у вигляді вкладених рівнів, де кожен наступний рівень є частиною ширшої системи (рис 1.1) [1].

Тестування (Testing) – це операційна діяльність, що полягає у виконанні тест-кейсів, перевірці роботи функціоналу та пошуку дефектів. Його основна мета – виявити невідповідності між фактичними та очікуваними результатами та зафіксувати помилки.

Контроль якості (Quality Control, QC) є ширшим процесом, що включає тестування, але охоплює також інспекції, перевірку відповідності продукту вимогам, аналіз дефектів та встановлення процедур їх усунення. QC фокусується на оцінюванні якості створеного продукту.

Забезпечення якості (Quality Assurance, QA) – це найвищий рівень системи управління якістю, спрямований на попередження появи дефектів. QA включає розробку стандартів, методик, процесів, політик та регламентів, які гарантують, що робота команди виконується структуровано, послідовно та відповідно до встановлених правил.

Важливо підкреслити, що тестування не гарантує повної відсутності помилок, але істотно зменшує ймовірність їх появи у фінальному продукті. Результати виконання тестів допомагають формувати об’єктивне уявлення про поточний стан продукту, прогнозувати ризики та ухвалювати рішення щодо подальших етапів розробки.

Рисунок 1.1 – Співвідношення Testing, Quality Control та Quality Assurance

Завдяки такому підходу забезпечення якості стає системним процесом, який охоплює як практичну перевірку програмного забезпечення, так і організацію методів, що гарантують стабільність, передбачуваність і відповідність кінцевого продукту вимогам.

1.2 Тестування у життєвому циклі розроблення програмного забезпечення

Життєвий цикл програмного забезпечення (SDLC) охоплює сукупність етапів, які проходить програмний продукт – від формування вимог до його впровадження та подальшої підтримки. Тестування є складовою більшості цих етапів, оскільки саме на стадії перевірки визначаються якість реалізованих функцій, відповідність вимогам та готовність системи до практичного використання [2].

У традиційних моделях розроблення, таких як водоспадна (Waterfall) та V-подібна модель, тестування виконується після завершення основних етапів проєктування та реалізації. Водоспадна модель передбачає послідовний рух від вимог до тестування, що робить виправлення помилок на пізніх етапах дорогим і трудомістким.

V-подібна модель підкреслює зв’язок між етапами розроблення та відповідними видами тестування: приймальне тестування співвідноситься з аналізом вимог, системне – із проєктуванням архітектури, інтеграційне – з проєктуванням компонентів. Такий підхід формує більш структуроване бачення ролі тестування у створенні ПЗ.

Для ітеративних та інкрементних моделей характерний циклічний процес, за якого тестування проводиться після кожної ітерації. Це дає змогу

поступово перевіряти окремі частини функціоналу, отримувати швидший зворотний зв’язок і зменшувати ризики накопичення критичних дефектів.

У сучасній індустрії більшість компаній використовують Agile-методології, такі як Scrum або Kanban, де тестування відбувається безперервно та інтегровано в кожен етап розроблення. У Scrum, наприклад, команда працює спринтами тривалістю 1–4 тижні, і протягом кожного спринту тестувальники перевіряють створений функціонал одразу після його реалізації. Це дозволяє оперативно виявляти помилки, швидко реагувати на зміну вимог і підтримувати стабільність продукту.

Загальний принцип сучасного підходу можна зобразити у вигляді циклу, де розроблення та тестування виконуються паралельно (рис 1.2). Це сприяє підвищенню якості програмного забезпечення, скороченню часу випуску релізів та мінімізації ризику появи критичних дефектів на етапі експлуатації.

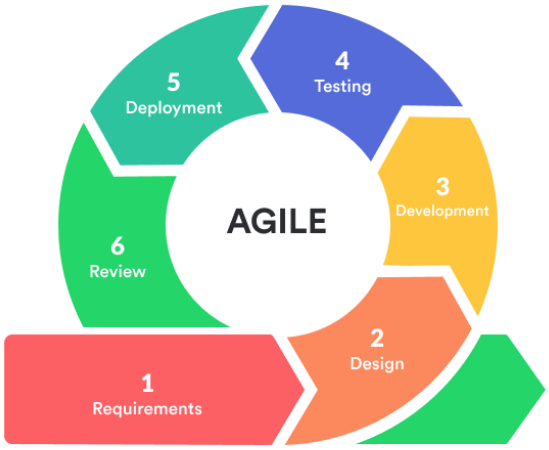


Рисунок 1.2 – Узагальнений Agile життєвий цикл

Таким чином, незалежно від обраної моделі, тестування відіграє ключову роль у забезпеченні якості та надійності продукту, виконуючи функції оцінювання, контролю та верифікації на кожному етапі його створення.

1.3 Типи та рівні тестування програмного забезпечення

Тестування програмного забезпечення охоплює різноманітні підходи, що відрізняються за рівнем деталізації, ціллю та способом взаємодії з системою. Відповідно до ISTQB Foundation Level, класифікація тестування може здійснюватися за декількома ознаками: рівнями виконання, функціональною спрямованістю, нефункціональними характеристиками, знанням внутрішньої структури та характером змін у програмному продукті.

1.3.1 Рівні тестування

Рівні тестування – це структурований підхід до організації тестового процесу, який дозволяє протестувати програмне забезпечення на різних стадіях розробки [2]. Кожен рівень спрямований на перевірку певних аспектів продукту та має визначені завдання. Загальна мета рівнів тестування полягає у виявленні дефектів і проблем якнайраніше, щоб забезпечити їх виправлення ще до виходу програмного продукту на ринок.

Існують наступні чотири рівня:

- Unit testing (модульне тестування) – перевіряє роботу окремих модулів або функцій програми з метою виявлення дефектів на ранніх етапах.
- Integration testing (інтеграційне тестування) – перевіряє взаємодію між модулями, правильність обміну даними та узгодженість компонентів.
- System testing (системне тестування) – оцінює працездатність програмного забезпечення як єдиної системи, перевіряючи функціональність, стабільність та відповідність вимогам.

- Acceptance testing (приймальне тестування) – фінальний рівень, що підтверджує готовність продукту до впровадження та відповідність бізнес-вимогам.

Пірамідальна структура рівнів тестування узагальнює логіку їх застосування та демонструє залежність між обсягом і глибиною виконуваних перевірок (рис 1.3).

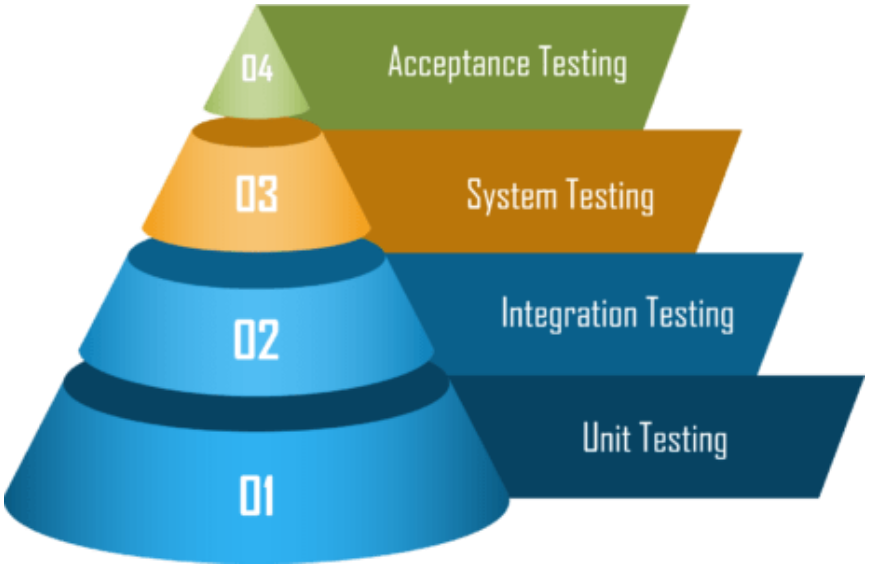


Рисунок 1.3 – Піраміда тестування

1.3.2 Функціональне тестування

Функціональне тестування є одним із найважливіших етапів процесу тестування програмного забезпечення. Його основна мета - перевірити, чи виконує програмний продукт всі функції та операції згідно з вимогами та специфікаціями. Під час функціонального тестування перевіряється здатність програми вірно обробляти вхідні дані, генерувати правильні результати та взаємодіяти з іншими компонентами системи.

Цей вид тестування спрямований на виявлення дефектів у логіці програми, відсутності або некоректності функцій, а також некоректного поведінки системи в різних умовах використання. Функціональне тестування допомагає переконатися, що програмний продукт відповідає вимогам бізнесу та очікуванням користувачів.

1.3.3 Нефункціональне тестування

Нефункціональне тестування є важливою складовою процесу перевірки програмного забезпечення, на відміну від функціонального тестування, яке перевіряє, чи відповідає програмний продукт функціональним вимогам. Не функціональне тестування спрямоване на оцінку якості програмного продукту та його властивостей, які не пов'язані безпосередньо з його функціональністю.

В рамках не функціонального тестування перевіряються характеристики програмного продукту, такі як продуктивність, безпека, надійність, масштабованість, сумісність, доступність, ефективність, зручність для користування та інші. Ці аспекти грають критичну роль у забезпеченні якості та задоволення користувачів. Основні типи нефункціонального тестування наведено у таблиці 1.1

Таблиця 1.1 Основні типи нефункціонального тестування

Тип	Опис	Приклад
Performance testing	Тестування продуктивності включає в себе вимірювання швидкості, ефективності та стабільності програмного продукту під час різних навантажень.	Перевірка часу завантаження сторінок веб-додатку, тестування роботи системи під високим навантаженням перевірка стійкості сервера під піковими запитами.
Usability testing	Цей тип тестування спрямований на оцінку того, наскільки легко та зручно користувачі можуть взаємодіяти з програмним продуктом.	Тестування зручності користування веб-сайту для онлайн-покупок, перевірка чіткості та зрозумілості інтерфейсу додатку.
Security testing	Тестування безпеки спрямоване на виявлення потенційних вразливостей та захисних проблем в програмному продукті з метою запобігання несанкціонованого доступу.	Перевірка реакції системи на недійсний ввід, перевірка прав доступу користувачів до конфіденційної інформації, атаки на веб-сайт, тестування на XSS або SQL ін'єкції.
Compatibility testing	Цей тип тестування перевіряє, як програмний продукт працює на різних операційних системах, пристроях та конфігураціях.	Тестування додатку на різних версіях операційної системи, перевірка взаємодії додатку на різних браузерах.
Configuration testing	Тестування конфігурації перевіряє, чи працює програмний продукт з правильними налаштуваннями та унікальними конфігураціями.	Тестування додатку з різними налаштуваннями (мова, часовий пояс, регіональні налаштування).

Тип	Опис	Приклад
UI testing	Тестування користувацького інтерфейсу перевіряє коректність та зручність взаємодії користувача з програмою через інтерфейс.	Візуальна перевірка графічних елементів (кнопки, меню, форми).
Localization/ Internationalization testing	Цей тип оцінює здатність програмного продукту використовувати міжнародні стандарти та мовні ресурси.	Перевірка підтримки різних мов та кодувань, локалізовані версії додатку для різних країн.
Accessibility testing	Тестування доступності перевіряє, наскільки доступний програмний продукт для людей з обмеженими можливостями або інвалідів.	Тестування можливості користувачів із зниженим зором, перевірка читабельності контенту для слабозорих.

Наведені типи нефункціонального тестування охоплюють основні аспекти оцінювання якості програмного забезпечення та найчастіше застосовуються під час перевірки веб-додатків. Однак перелік нефункціональних тестів може буде значно більшим. Конкретний набір типів тестування визначається архітектурою продукту, вимогами до якості та особливостями програмної системи.

1.3.4 Тестування за рівнем знань про систему

Тестування за рівнем знань про внутрішню структуру системи класифікує підходи до перевірки програмного забезпечення залежно від того, наскільки тестувальник знайомий із внутрішньою реалізацією продукту. Виділяють три основні підходи: тестування методом чорного ящика, білого ящика та сірого ящика.

Тестування чорного (Black-box testing) ящика ґрунтується на перевірці програмного продукту без знання його внутрішньої структури або реалізації. У цьому підході тестувальник працює виключно з зовнішніми інтерфейсами, вхідними даними та очікуваними результатами згідно зі специфікацією. Згідно з ISTQB, тестування методом чорного ящика може бути як функціональним, так і нефункціональним, та використовується на всіх рівнях: модульному, інтеграційному, системному й приймальному. Основна ідея цього підходу полягає в тому, що тестування базується на вимогах та очікуваній поведінці системи, а не на аналізі її коду. Це дозволяє оцінити продукт так само, як це робив би кінцевий користувач.

Тестування білого ящика (White-box testing) передбачає повний доступ до внутрішньої структури, коду та логіки програми. Тестувальник аналізує алгоритми, шляхи виконання, умови переходів і структури даних, створюючи тестові сценарії з урахуванням реальної реалізації. Метод застосовується на різних рівнях, але найбільш типово – під час модульного тестування або створення unit-тестів розробниками чи інженерами SDET. По суті, тестування білого ящика є перевіркою шляхів виконання коду та його внутрішньої логіки.

Метод сірого ящика (Grey-box testing) поєднує властивості чорного та білого ящика. Тестувальник має часткове знання про внутрішню структуру системи (наприклад, документацію, діаграми, специфікації API), але сам процес тестування виконується з позиції зовнішнього користувача. Хоча метод сірого ящика не входить до офіційної класифікації ISTQB, він широко використовується на практиці. Часткове знання внутрішньої структури дозволяє створювати більш ефективні тестові сценарії, поглиблювати покриття та швидше знаходити причини дефектів. Прикладами gray-box тестування є перевірка коректності записів у базі даних, аналіз логів, перегляд внутрішніх метрик або моніторинг станів системи.

1.3.5 Тестування після змін

У процесі розроблення програмного забезпечення постійно відбуваються зміни, пов’язані з додаванням нового функціоналу, виправленням дефектів, оновленням інтерфейсів або модифікацією бізнес-логіки. Кожна зміна може вплинути на роботу існуючих компонентів, спричинити появу нових помилок або порушити вже стабільний функціонал. Тому тестування після змін є важливою частиною забезпечення якості та стабільності програмного продукту.

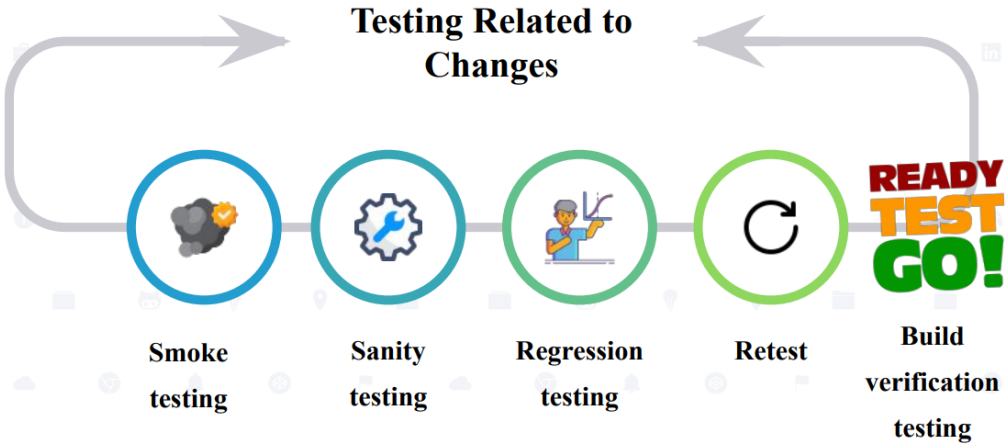


Рисунок 1.4 – Основні типи тестування, пов’язані зі змінами у системі

Показані на рисунку типи тестування виконуються після внесення змін до програмного продукту та доповнюють один одного, забезпечуючи всебічну перевірку системи.

Regression testing – повторне виконання тестів для підтвердження того, що нові зміни не порушили роботу попередньо реалізованого функціоналу. Регресійне тестування виконується після кожного релізу або суттєвого оновлення системи і є одним із ключових елементів підтримки стабільності програмного продукту.

Retest – цілеспрямована перевірка виправлених дефектів. На відміну від регресії, retest оцінює успішність усунення конкретної помилки, перевіряючи ті самі сценарії, які раніше призводили до збою.

Smoke testing – базова перевірка критично важливого функціоналу після отримання нової збірки або оновлення системи. Метою smoke testing є визначення загальної працездатності продукту та можливості продовження подальших етапів тестування.

Sanity testing – локалізована, спрощена форма перевірки, що виконується для підтвердження коректності роботи окремого елемента після внесення змін саме до нього. Sanity testing забезпечує швидке підтвердження того, що конкретний модуль працює згідно з вимогами.

У сукупності ці види тестування дозволяють контролювати вплив змін на систему, запобігати виникненню регресійних помилок і гарантувати стабільність програмного забезпечення на кожному етапі його розвитку.

1.4 Процес тестування програмного забезпечення

Тестування програмного забезпечення здійснюється за визначеною послідовністю етапів, що разом утворюють життєвий цикл тестування (Software Testing Life Cycle, STLC). Чітка структура процесу забезпечує передбачуваність результатів, контроль якості та своєчасне виявлення дефектів. Нижче подано основні стадії STLC.

1. Аналіз вимог

На цьому етапі вивчаються функціональні й нефункціональні вимоги, визначаються пріоритети, ризики та обсяг майбутнього тестування. Аналіз вимог формує розуміння того, що саме потрібно протестувати і які критерії якості застосовуються до продукту.

2. Планування тестування

Передбачає визначення стратегії тестування, вибір підходів та інструментів, оцінювання ресурсів, строків і відповідальних осіб. Результатом є тестовий план, який визначає рамки та правила проведення тестування.

3. Проєктування тестів

На цьому етапі створюються тест-кейси, сценарії, чек-листи, моделюються тестові дані. Тест-дизайн визначає, як саме буде перевірятися функціонал і які умови будуть використані під час тестування.

4. Підготовка тестового середовища

Включає налаштування інфраструктури, серверів, баз даних, конфігурацій додатку, користувацьких облікових записів та всіх необхідних компонентів. Коректно підготовлене середовище забезпечує відтворюваність результатів тестування.

5. Виконання тестів

На цьому етапі тестувальники виконують створені сценарії, порівнюють фактичні результати з очікуваними, фіксують дефекти та відстежують їхній статус. У межах цього етапу також проводиться повторне тестування після виправлення помилок та, за потреби, регресійне тестування.

6. Завершення тестування та звітність

Підсумовуються результати тестування, оцінюється ступінь покриття вимог, визначається загальний рівень якості продукту та готуються звіти. На основі отриманих даних приймається рішення щодо готовності програмного забезпечення до релізу.

STLC забезпечує логічну та впорядковану організацію процесу тестування. Кожний етап виконує окрему функцію, а разом вони формують

завершений цикл, який дозволяє контролювати якість програмного продукту та мінімізувати ризики появи критичних дефектів на етапі експлуатації.

1.5 Метрики та показники якості процесів тестування

Основна мета тестування полягає у наданні достовірної інформації для прийняття рішень і управління ризиками. Щоб контролювати хід тестування та оцінювати якість програмного продукту, використовуються метрики – кількісні показники, які характеризують як процес тестування, так і результати виконаних перевірок. Метрики дозволяють визначити ефективність підходів, відстежувати динаміку якості та забезпечувати прозорість для зацікавлених сторін.

Загалом виділяють дві основні групи метрик: метрики процесу та метрики продукту.

Метрики процесу застосовуються для оцінювання організації й ефективності тестового процесу. До них належать показники продуктивності підготовки та виконання тестів, охоплення тестами вимог, а також успішності виконання тест-кейсів. Такі метрики дозволяють визначити темп роботи команди, контролювати обсяг виконаних перевірок, аналізувати відхилення від плану та оптимізувати процес розроблення тестової документації.

Окреме місце посідають метрики виконання тестів, що відображають частку успішних, провалених або заблокованих тест-кейсів. Ці показники допомагають оцінити стабільність системи на поточному етапі та своєчасно визначити критичні модулі чи сценарії, що потребують додаткової уваги.

Метрики продукту спрямовані на оцінку якості програмного забезпечення за кількістю та характером дефектів. До ключових належать рівень виявлення помилок, щільність дефектів, ефективність їх виправлення та

частка дефектів, які залишилися непоміченими під час тестування. Показники на кшталт Defect Detection Percentage або Defect Removal Efficiency дозволяють оцінити, наскільки успішно тестування покриває ризики і наскільки якісно команда усуває знайдені проблеми.

Важливим видом метрик є тестове покриття, яке визначає, наскільки повно протестовано функціональні вимоги, код або користувацькі сценарії. До покриття можуть належати вимоги (requirements coverage), покриття коду (code coverage) чи покриття шляхів виконання. Жоден із цих підходів не є абсолютним, адже покриття коду не виявляє відсутні вимоги, а покриття вимог може не враховувати нетривіальні випадки, що виникають через реальну реалізацію. У зв’язку з цим тестове покриття розглядається як допоміжний показник, який використовується разом з іншими метриками.

У цілому використання метрик забезпечує контрольованість процесу тестування та підвищує якість програмного забезпечення. Вони дозволяють своєчасно ідентифікувати проблеми, оцінити стабільність продукту та сформувати обґрунтовані рекомендації щодо готовності системи до релізу.

1.6 Висновки до першого розділу

У першому розділі було розглянуто теоретичні основи тестування програмного забезпечення, а також визначено його місце у життєвому циклі розроблення ПЗ. Було показано, що тестування є ключовим інструментом забезпечення якості та суттєво впливає на зниження ризиків, пов’язаних з випуском програмного продукту.

Аналіз життєвого циклу розроблення продемонстрував, що тестування супроводжує продукт на всіх етапах його створення, а сучасні підходи, зокрема Agile, передбачають тісну інтеграцію процесів тестування і розроблення. Розглянута класифікація типів і рівнів тестування дала змогу систематизувати

функціональні, нефункціональні та структурні підходи до перевірки якості програмних систем.

Окрему увагу приділено методам тестування за рівнем знань про систему, що дозволяють здійснювати перевірку як з позиції користувача, так і з урахуванням внутрішньої структури програмного продукту. Описані також особливості тестування після змін, що є важливим для підтримання стабільності системи в умовах її постійного удосконалення.

Теоретичний розділ сформував комплексне підґрунтя для подальшого аналізу специфіки тестування веб-додатків та проведення практичного дослідження у наступних розділах.

2 ВЕБ-ДОДАТКИ ЯК ОБ’ЄКТ ТЕСТУВАННЯ

У першому розділі були розглянуті загальні підходи до тестування ПЗ, його цілі, принципи та види. Однак вибір конкретних методів тестування значною мірою залежить від типу програмної системи. Одним із найпоширеніших і водночас найскладніших об’єктів тестування сьогодні є веб-додатки, що працюють у клієнт-серверному середовищі та забезпечують доступ до сервісів через Інтернет.

Веб-додатки стали основою для більшості сучасних сервісів – від інтернет-магазинів і банківських систем до навчальних платформ та державних порталів. Їх переваги (доступ з будь-якого пристрою, відсутність потреби в інсталяції, централізовані оновлення) поєднуються з додатковими ризиками: залежністю від мережі, різноманітністю браузерів та пристроїв, підвищеними вимогами до безпеки й продуктивності.

У цьому розділі веб-додатки розглядаються як специфічний об’єкт тестування. Послідовно аналізуються їх архітектурні особливості, класифікація, модель якості та напрями тестування, що надалі дозволяє обґрунтувати вибір методів практичного дослідження.

2.1 Поняття веб-додатків та їх архітектурні особливості

Веб-додаток – це програмна система, що функціонує за моделлю клієнт–сервер і забезпечує користувачеві доступ до функціональності через веб-браузер [7]. Користувач працює з інтерфейсом у браузері (клієнтська частина), а обробка запитів, бізнес-логіки та доступ до баз даних здійснюється на сервері.

На відміну від класичних настільних програм, веб-додатки:

- не потребують встановлення на пристрій користувача;
- можуть оновлюватися централізовано на сервері;
- доступні з різних платформ (ПК, планшети, смартфони).

Більшість сучасних веб-додатків будується за багаторівневою архітектурою (рис 2.1) [13]. Типова трирівнева модель включає:

1. Клієнтський рівень (Presentation tier)

Відповідає за відображення інтерфейсу та взаємодію з користувачем. Реалізується засобами HTML, CSS, JavaScript та фронтенд-фреймворками (React, Angular, Vue.js тощо).

2. Рівень логіки (Application / Logic tier)

Реалізує бізнес-правила, обробляє запити з клієнта, викликає зовнішні сервіси й API. Типові технології це Node.js, Django, Spring, ASP.NET тощо.

3. Рівень даних (Data tier)

Забезпечує зберігання та доступ до інформації в базах даних (MySQL, PostgreSQL, MongoDB, Redis та ін.).

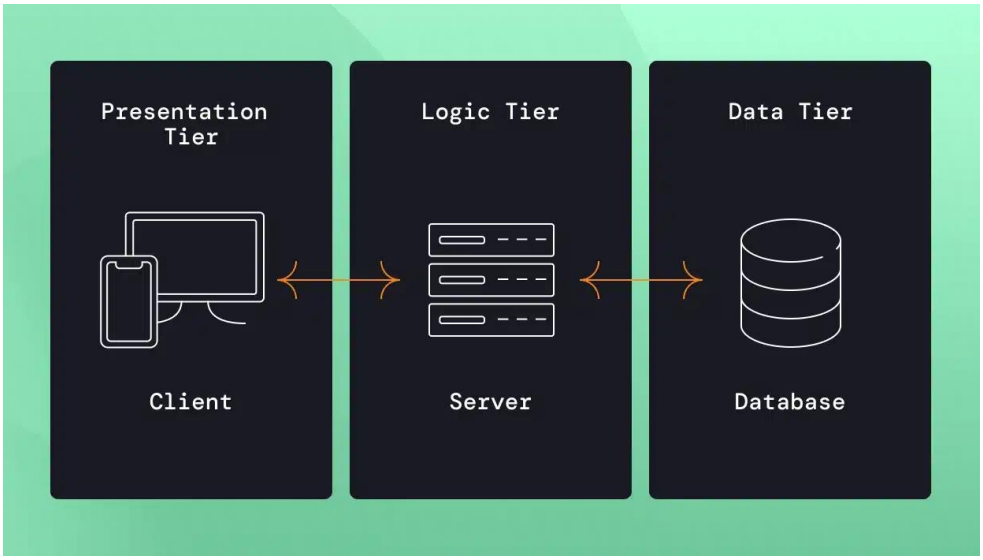


Рисунок 2.1 – Типова трирівнева архітектура веб-додатку

Така структура спрощує масштабування та супровід, але ускладнює тестування: перевіряти потрібно не лише окремі рівні, а й коректність їх взаємодії, узгодженість форматів даних, стабільність API та поведінку під навантаженням.

Враховуючи наведені архітектурні характеристики, веб-додатки мають низку специфічних властивостей, які безпосередньо впливають на організацію та обсяг тестування. Для узагальнення ключових факторів, що визначають складність та особливості перевірки веб-систем, іх наведено у таблицю 2.1, в якій систематизовано основні характеристики веб-додатків як об’єктів тестування.

Таблиця 2.1 – Основні особливості веб-додатків як об’єктів тестування

Особливість веб-додатка	Опис	Вплив на тестування
Розподілена клієнт–серверна архітектура	Взаємодія між клієнтською частиною, сервером та базами даних через мережеві протоколи (HTTP/HTTPS).	Необхідність тестування мережевих запитів, API, коректності обробки помилок, стабільності комунікації.
Динамічний контент (SPA, AJAX)	Інтерфейс оновлюється без перезавантаження сторінки; активне використання JavaScript.	Складніше тестування станів UI, необхідність перевірки асинхронних операцій, правильної обробки подій.
Багатокористувацький доступ	Одночасна робота великої кількості користувачів.	Потребує навантажувального, стресового та об’ємного тестування.

Особливість веб-додатка	Опис	Вплив на тестування
Кросбраузерність та кросплатформність	Робота на різних пристроях, ОС та браузерах (Chrome, Firefox, Safari тощо).	Вимагає тестування відображення, поведінки та сумісності на різних комбінаціях середовищ.
Підвищені вимоги до безпеки	Дані передаються через відкриті мережі; можливість атак (XSS, CSRF, SQLi).	Проведення security testing, перевірка авторизації, шифрування, політик доступу, обробки чутливих даних.
Залежність від продуктивності мережі та сервера	Доступність додатку визначається якістю мережі та ресурсами серверів.	Тестування швидкодії (Performance), стабільності під навантаженням, latency, timeouts.
Масштабованість та еластичність	Додаток має витримувати зростання кількості користувачів та операцій.	Необхідність Scalability testing, аналізу поведінки системи під збільшенням навантаження.

Як видно з таблиці 2.1, ключові властивості веб-додатків формують широкий спектр вимог до процесу тестування, який має охоплювати не лише функціональну перевірку, а й оцінку продуктивності, безпеки, сумісності та зручності використання. Урахування цих характеристик є необхідною умовою для забезпечення якості розробки та стабільної роботи веб-додатків у реальних умовах експлуатації.

2.2 Класифікація веб-додатків

У сучасній практиці веб-розроблення веб-додатки класифікують за їх структурою, поведінкою та способом взаємодії з користувачем. Нижче наведено один із найбільш поширених та загальновизнаних підходів до класифікації веб-додатків, що використовується у професійній літературі та індустріальних оглядах [8].

1. Статичні веб-додатки (Static Web Applications)

Простий тип веб-додатків, у яких контент залишається незмінним без втручання розробника. Сторінки формуються заздалегідь і відображаються користувачам у незмінному вигляді.

2. Динамічні веб-додатки (Dynamic Web Applications)

Контент таких систем генерується сервером під час виконання запиту. Сторінки змінюються залежно від даних, стану системи та дій користувача.

3. Односторінкові веб-додатки (Single Page Applications, SPA)

SPA завантажуються один раз, після чого працюють через динамічне оновлення інтерфейсу без перезавантаження сторінки. Інтенсивно використовують JavaScript та API.

4. Багатосторінкові веб-додатки (Multi-Page Applications, MPA)

Традиційна модель, у якій кожен запит користувача призводить до завантаження окремої сторінки. Підходить для великих систем із багатим функціоналом.

5. E-commerce веб-додатки (E-commerce Web Applications)

Спеціалізовані системи, що забезпечують продаж товарів та послуг. Містять каталоги, кошики, платіжні модулі та модулі управління замовленнями.

6. Портальні веб-додатки (Portal Web Applications)

Системи, що агрегують різні сервіси або дані в одному інтерфейсі: кабінети користувача, корпоративні портали, освітні платформи.

7. Анімовані веб-додатки (Animated Web Applications)

Використовують інтерактивну графіку, анімацію та динамічні ефекти. Часто створюються для промо-сторінок, презентацій та креативних проєктів.

8. CMS-орієнтовані веб-додатки (CMS-based Applications)

Побудовані на основі систем керування контентом (WordPress, Drupal, Joomla). Використовуються для блогів, корпоративних сайтів, інформаційних ресурсів.

9. Прогресивні веб-додатки (Progressive Web Applications, PWA)

Технологія веб-додатків, яка поєднує можливості вебу та мобільних застосунків: офлайн-режим, push-сповіщення, кешування, встановлення на пристрій.

Ця класифікація дає змогу систематизувати різні типи веб-додатків, враховуючи їхню поведінку, спосіб рендерингу, інструменти реалізації та функціональне призначення [9]. Такий підхід дозволяє точніше визначити вимоги до тестування та специфіку перевірки кожного типу веб-систем.

2.3 Параметри якості веб-додатків

Якість веб-додатку визначається його здатністю стабільно виконувати передбачені функції, забезпечуючи коректну та передбачувану взаємодію користувача з системою в різних умовах експлуатації. У міжнародній практиці

оцінювання якості програмного забезпечення ґрунтується на стандарті ISO/IEC 25010:2016, який визначає модель якості ПЗ через вісім ключових характеристик [1]: функціональну придатність, ефективність (продуктивність), сумісність, зручність використання, надійність, безпеку, супроводжуваність та портативність.

Рисунок 2.2 – Модель характеристик якості програмного забезпечення за ISO/IEC 25010:2016

Після наведеної моделі нижче представлено детальний опис кожної характеристики окремо, з урахуванням специфіки веб-додатків.

Функціональна придатність (Functional Suitability)

Ця характеристика визначає, наскільки веб-додаток відповідає функціональним вимогам користувачів і виконує необхідні бізнес-завдання.

Основні атрибути:

- повнота функцій – реалізовані всі необхідні можливості;

- коректність результатів – логіка працює передбачувано та дає правильні результати;
- відповідність вимогам – функціонал відповідає специфікаціям.

Оцінювання здійснюється переважно методами функціонального тестування, включно з розробкою тест-кейсів для кожної вимоги.

Ефективність (Performance Efficiency)

Ефективність характеризує швидкість реакції веб-додатку, стабільність роботи під навантаженням і оптимальність використання ресурсів.

У веб-середовищі кількісне оцінювання продуктивності проводиться згідно з метриками Google Web Vitals, які вимірюють швидкість відображення сторінки, стабільність інтерфейсу та затримку першої взаємодії (TTFB, LCP, FID, CLS, TBT) [5].

Сумісність (Compatibility)

Сумісність визначає здатність веб-додатку коректно працювати на різних платформах, у різних браузерах та пристроях.

Оцінюється через:

- кросбраузерне тестування;
- кросбраузерне тестування;
- перевірку коректності взаємодії з іншими сервісами та API.

Зручність використання (Usability)

Usability характеризує рівень інтуїтивності та зрозумілості інтерфейсу, а також простоту виконання користувачами своїх цілей [10].

Основні аспекти оцінювання:

- зрозумілість інтерфейсу;
- послідовність елементів;
- наявність зворотного зв’язку;

- профілактика помилок користувача.

Тестування проводиться за допомогою евристичного аналізу, юзабіліті-тестування або A/B-тестування.

Надійність (Reliability)

Надійність визначає здатність веб-додатку стабільно функціонувати в умовах навантаження, відмов чи непередбачених ситуацій.

Ключові підпараметри:

- відновлюваність (recovery);
- доступність сервісу (uptime);
- здатність обробляти помилки без втрати даних.

Безпека (Security)

Безпека – одна з найважливіших характеристик для сучасних веб-систем, адже вона визначає здатність додатку захищати дані користувача від несанкціонованого доступу, атак і витоків.

Оцінювання безпеки ґрунтується на рекомендаціях OWASP Top 10, включно з перевіркою [3]:

- захисту авторизації та автентифікації;
- управління сесіями;
- оброблення вхідних даних;
- стійкості до XSS, SQL Injection тощо.

Супроводжуваність (Maintainability)

Супроводжуваність визначає простоту оновлення, модифікації чи усунення дефектів у веб-додатку.

Важливими факторами є:

- структурованість коду;

- наявність документації;
- покриття тестами;
- модульність архітектури.

Портативність (Portability)

Портативність характеризує можливість розгортання додатку в різних середовищах з мінімальними змінами конфігурації.

Для сучасних веб-рішень особливо важливими є:

- підтримка контейнеризації (Docker, Kubernetes);
- можливість роботи в хмарних середовищах (AWS, Azure, Google Cloud);
- гнучкість масштабування.

Комплексне оцінювання якості веб-додатків за моделлю ISO/IEC 25010, у поєднанні з практичними метриками Google Web Vitals та рекомендаціями OWASP, дозволяє об’єктивно визначити рівень надійності, безпеки й продуктивності сучасних веб-систем.

2.4 Особливості тестування веб-додатків

Тестування веб-додатків має низку специфічних характеристик, що відрізняють його від перевірки традиційного програмного забезпечення. Це обумовлено архітектурою веб-систем, наявністю багатьох взаємодіючих компонентів, різноманітністю середовищ доступу, постійними оновленнями та необхідністю забезпечення стабільності роботи в умовах значної кількості користувачів.

Основна мета тестування веб-додатку полягає у підтвердженні того, що система коректно виконує задані функції, залишається доступною, безпечною та зручною для користувачів у реальних мережевих умовах.

2.4.1 Технологічні чинники, що впливають на тестування веб-додатків

На відміну від настільних програм, веб-додаток функціонує у розподіленому середовищі та включає клієнтську, серверну й мережеву частини. Це створює додаткові виклики для тестування, зокрема:

1. Клієнт–серверна взаємодія.

Потребує перевірки коректності обміну даними між фронтендом і бекендом, правильності REST/GraphQL-запитів, обробки помилок і форматів відповідей API.

2. Мережеві фактори.

Продуктивність системи залежить від швидкості Інтернет-з’єднання, затримок, кешування та доступності віддалених ресурсів. Необхідно враховувати сценарії низької пропускної здатності, нестабільного каналу й офлайн-режиму (PWA).

3. Браузерна залежність.

Через відмінності в рендерах браузерів (Chromium, Gecko, WebKit) виникає потреба в кросбраузерному тестуванні та перевірці адаптивної верстки.

4. Часті оновлення.

Більшість сучасних веб-проєктів використовує CI/CD, що передбачає регулярні релізи. Це посилює потребу в регресійному тестуванні після кожної зміни.

5. Безпека.

Веб-додатки працюють у відкритій мережі, що робить їх вразливими до атак типу SQL Injection, XSS, CSRF, SSRF тощо. Тестування безпеки є обов’язковою частиною процесу.

2.4.2 Основні напрями тестування веб-додатків

Тестування веб-додатків охоплює низку взаємопов’язаних напрямів, кожен із яких спрямований на перевірку окремих характеристик системи.

Нижче у таблиці 2.2 подано основні напрямів тестування веб-додатків.

Таблиця 2.2 – Основні напрями тестування веб-додатків

Напрям тестування	Мета	Приклади інструментів	Особливості реалізації
API-тестування	Перевірка коректності обміну даними між клієнтом і сервером	Postman, Swagger, Newman	Тестування REST/GraphQL-запитів, статус-кодів, форматів JSON
Навантажувальне тестування	Оцінка стабільності й швидкодії під високим навантаженням	Apache JMeter, Gatling	Моделювання одночасних запитів, вимір часу відгуку, визначення «вузьких місць»
Безпекове тестування	Виявлення вразливостей, що загрожують даним користувачів	OWASP ZAP, Burp Suite	Перевірка авторизації, керування сесіями, конфігурацій та обробки вхідних даних

Напрям тестування	Мета	Приклади інструментів	Особливості реалізації
Юзабіліті-тестування	Оцінка зручності інтерфейсу та користувацьких сценаріїв	Hotjar, Lookback, Maze	Евристичний аналіз, дослідження поведінки користувачів
Тестування доступності (Accessibility)	Перевірка відповідності WCAG 2.1	axe, WAVE	Аналіз ARIA-атрибутів, контрастності, навігації з клавіатури

2.4.3 Організаційні та технічні аспекти тестування

Веб-тестування тісно інтегроване у процес розроблення, що відповідає концепціям DevOps і Continuous Testing.

Ключові аспекти:

- CI/CD-конвеєри забезпечують автоматичний запуск тестів після кожного коміту або релізу.
- Контейнеризація (Docker, Kubernetes) гарантує стабільність тестового середовища.
- Моніторинг та логування (Grafana, Kibana, Prometheus) дозволяють виявляти дефекти, що проявляються лише в продакшені.
- Віртуалізація тестових даних та API (Mock Server, WireMock) дає змогу тестувати систему без доступу до реальних сервісів.

Автоматизовані тести скорочують час регресійної перевірки на 40–60 %, знижуючи ризик людських помилок і прискорюючи випуск релізів.

Особливості тестування веб-додатків визначаються складною архітектурою, постійною взаємодією з мережею, а також високими вимогами до безпеки, продуктивності та зручності користування. Для забезпечення якості необхідно поєднувати функціональні, нефункціональні та автоматизовані перевірки на всіх рівнях тестування.

2.5 Висновки до другого розділу

У цьому розділі було розглянуто ключові особливості веб-додатків, їх класифікацію, параметри якості та специфіку процесів тестування у веб-середовищі. Веб-додатки характеризуються розподіленою архітектурою, багаторівневою структурою, залежністю від мережевих ресурсів і великою варіативністю середовищ виконання. Це зумовлює унікальні вимоги до забезпечення їхньої надійності, безпеки та продуктивності.

Класифікація веб-додатків за архітектурними, технологічними та функціональними критеріями дозволила визначити основні типи систем (статичні, динамічні, SPA, PWA, портальні, багаторівневі та мікросервісні рішення) та окреслити особливості їхнього використання і тестування. На основі моделі ISO/IEC 25010 було визначено ключові параметри якості веб-додатків, такі як функціональна придатність, продуктивність, сумісність, надійність, безпека, зручність використання, супроводжуваність та портативність. Кожен із цих параметрів формує вимоги до тестування та визначає відповідні підходи до перевірки систем.

Проведений аналіз підтверджує, що комплексне тестування веб-додатків потребує застосування різних методів і технік на всіх рівнях життєвого циклу, а також врахування архітектурних та технологічних особливостей системи. Отримані теоретичні результати слугуватимуть основою для практичного дослідження методів тестування, яке буде виконано у наступному розділі.

3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ (ЧАСТИНА 1)

Практична частина дипломної роботи спрямована на дослідження ефективності методів тестування веб-додатків на прикладі реального комерційного сервісу. Вибір об’єкта дослідження, методів тестування та послідовності перевірок дозволяє всебічно оцінити якість веб-додатку з позицій функціональності, зручності використання, продуктивності та сумісності.

У сучасній інженерії якості термін «методи тестування» використовується у широкому значенні, що узгоджується з підходами ISTQB Foundation [2]. На практиці він охоплює види тестування (test types), підходи тестування (manual, automated, exploratory) та техніки створення тестів.

У межах цієї роботи під методами тестування розглядаються насамперед види тестування, які були застосовані під час оцінювання веб-додатку Rozetka.ua: функціональне тестування, юзабіліті-тестування, тестування сумісності, тестування продуктивності та базові перевірки безпеки. Такий підхід дозволяє дослідити ефективність зазначених методів у реальних умовах експлуатації веб-системи.

3.1 Загальна характеристика веб-додатку Rozetka.ua

Rozetka.ua є одним із найбільш відвідуваних українських онлайн-ресурсів і функціонує як повноцінний веб-додаток електронної комерції. Його інтерфейс побудований таким чином, щоб забезпечити користувачеві повний цикл роботи з товаром: від пошуку та аналізу характеристик до здійснення

покупки та відстеження замовлення. Структура веб-додатку охоплює різноманітні елементи інтерфейсу та модулі, що утворюють комплексну систему користувацьких сценаріїв.

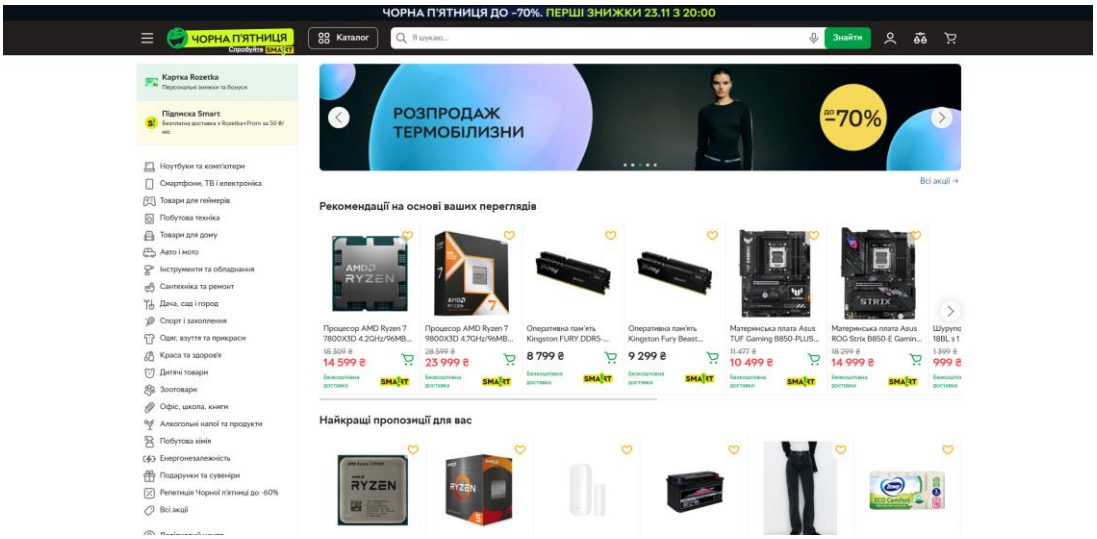


Рисунок 3.1 – Головна сторінка Rozetka.ua

Функціональна структура платформи включає:

- головну сторінку з навігаційним меню, категоріями товарів та рекламними блоками;
- каталог товарів із розвиненими фільтрами та сортуванням;
- сторінки товарів з детальним описом, фото, характеристиками, відгуками та рекомендованими товарами;
- функціонал кошика, оформлення замовлення та вибору способів доставки/оплати;
- персональний кабінет користувача.

До ключових користувацьких сценаріїв, що формують основу поведінки у системі, належать:

- пошук товарів;

- застосування фільтрів і сортування;
- перегляд характеристик і відгуків;
- додавання товарів до кошика та редагування його вмісту;
- оформлення замовлень;
- робота з особистим кабінетом.

Ці сценарії охоплюють основні аспекти електронної комерції, що робить Rozetka.ua репрезентативною платформою для дослідження методів тестування.

3.2 Обґрунтування вибору методів тестування

Вибір методів тестування визначався функціональною складністю веб-додатку, кількістю інтерактивних елементів, різноманітністю користувацьких сценаріїв та вимогами до продуктивності й надійності.

У межах роботи були обрані такі методи тестування (види), що дозволяють комплексно оцінити якість веб-додатку:

1. Функціональне тестування – для перевірки коректності ключових бізнес-функцій та користувацьких сценаріїв.
2. Юзабіліті-тестування – для оцінки інтуїтивності інтерфейсу, логічності навігації та зручності виконання дій.
3. Тестування сумісності – для перевірки роботи веб-додатку у різних браузерах та на різних пристроях.
4. Тестування продуктивності – для оцінки швидкодії та стабільності роботи під час взаємодії користувача з інтерфейсом.
5. Базові перевірки безпеки – для виявлення потенційних загроз, пов’язаних з доступністю персональних даних та коректністю обробки клієнтських запитів.

Таке поєднання методів забезпечує реалістичне наближення до практики тестування сучасних веб-систем та дозволяє дослідити роль кожного виду тестування у забезпеченні якості.

3.3 Проведення фунціонального тестування

Проведення тестування включало:

- опрацювання ключових сценаріїв користувача;
- проєктування чек-листа для поверхневої функціональної перевірки;
- створення двох поглиблених тест-кейсів для аналізу критичних зон;
- фіксацію результатів перевірок та виявлених дефектів;
- формування висновків щодо якості веб-додатку.

3.3.1 Чек-лист функціонального тестування

Функціональне тестування проводилося з метою перевірки коректності роботи основних бізнес-процесів: пошуку товарів, застосування фільтрів, сортування, перегляду товарів, роботи кошика та оформлення замовлення.

Було розроблено та виконано чек-лист, що охоплює основні сценарії взаємодії користувача з платформою. Результати чек-листа наведено у таблиці 3.1.

Таблиця 3.1 – Чек-лист функціонального тестування веб-додатку Rozetka.ua

Id	Опис перевірки	Статус (P/F)	Коментар/Тестові дані
Пошук			
1.1	Перевірити доступність та видимість поля пошуку	P	Поле пошуку відображається у шапці сайту, доступне для введення, активується кліком
1.2	Виконати пошук за коректним запитом	P	Запит: «ноутбук». Знайдено 5680 товарів
1.3	Перевірити появу підказок під час введення	P	Під час введення запиту з’являються релевантні підказки
1.4	Виконати пошук за частковим запитом	P	Для запиту «qiowruq832» система не знайшла товарів та відобразила сторінку з повідомленням «За запитом нічого не знайдено»
1.5	Виконати пошук неіснуючого товару	P	Пошук за частковим запитом «ноут» повертає релевантні результати – ноутбуки різних брендів
1.6	Очистити поле пошуку	P	Поле пошуку успішно очищається
1.7	Перевірити роботу кнопки «Знайти»	P	Кнопка «Знайти» ініціює пошук коректно
1.8	Перевірити поведінку пошуку без введеного тексту	P	При натисканні кнопки «Знайти» без введеного тексту нічого не відбувається – запит не виконується, сторінка не оновлюється
Каталог і навігація			
2.1	Відкрити основну категорію через меню	P	Сторінка категорії відкривається коректно
2.2	Перейти в підкатегорію «Ноутбуки»	P	Підкатегорія відкривається коректно, відображається список товарів і фільтрів
2.3	Перевірити роботу хлібних крихт	P	Відображається навігаційний ланцюжок «Комп’ютери та ноутбуки → Ноутбуки».

Id	Опис перевірки	Статус (P/F)	Коментар/Тестові дані
			Перехід назад за крихтою працює коректно
2.4	Перегорнути сторінку каталогу (пагінація)	P	Пагінація парцює коректно
2.5	Повернутися назад з картки товару та зберегти позицію	P	Після відкриття товару та повернення назад список товарів зберігає попередню позицію скролу
2.6	Перевірити відображення кількості знайдених товарів	P	Верхня частина сторінки коректно показує кількість товарів у вибраній категорії
2.7	Перевірити перемикання режимів відображення (плитка/список)	P	Перемикання між режимами відображення (плитка / список) працює коректно, інтерфейс оновлюється без помилок
Фільтри			
3.1	Застосувати фільтр «Бренд → Apple»	P	Фільтр працює коректно, у списку залишаються лише товари бренду Apple
3.2	Застосувати фільтр за ціною	P	Введений діапазон цін застосовується правильно
3.3	Застосувати кілька фільтрів одночасно	P	При одночасному застосуванні кількох фільтрів результати відповідають обом критеріям
3.4	Перевірити коректність кількості товарів у фільтрах	P	Кількість товарів у фільтрі «Бренд» відповідає фактичній кількості відображених товарів після застосування фільтрів
3.5	Скинути всі фільтри	P	Після скидання фільтрів відображається повний список товарів категорії
3.6	Застосувати фільтр «В наявності»	P	Після застосування фільтра показуються лише товари зі статусом «Є в наявності»

Id	Опис перевірки	Статус (P/F)	Коментар/Тестові дані
3.7	Перевірити роботу пошуку всередині фільтрів (поле «Пошук у фільтрах»)	P	Поле пошуку у фільтрах працює коректно
Сортування			
4.1	Сортування «За ціною: дешевші»	P	Товари впорядковуються за зростанням ціни
4.2	Сортування «За ціною: дорожчі»	P	Сторінка коректно сортирує товари за спаданням ціни
4.3	Сортування «За рейтингом»	P	Результати впорядковуються відповідно до рейтингу
4.4	Перевірити відсутність конфлікту між сортуванням і фільтрами	P	Після застосування будь-якого фільтра сортування продовжує працювати коректно
Картка товару			
5.1	Відкрити картку товару	P	Картка товару відкривається коректно, інформація завантажується без помилок
5.2	Перемикати фото у галереї	P	Перемикання фото працює коректно: зображення змінюються, відсутні пусті або биті зображення
5.3	Перевірити відображення ціни	P	Ціна, стара ціна (якщо є знижка) відображаються коректно
5.4	Перевірити відображення статусу наявності	P	Статус наявності відображаються коректно
5.5	Перевірити кнопку «Купити»	P	Кнопка активна; натискання приводить до додавання товару у кошик
5.6	Перевірити вкладку характеристик	P	Усі основні характеристики відображаються, вкладка відкривається коректно
5.7	Перевірити вкладку відгуків	P	Контент завантажується, текст повністю відображається, скролинг працює коректно

Id	Опис перевірки	Статус (P/F)	Коментар/Тестові дані
5.8	Перевірити рекомендації/схожі товари	P	Відображаються рекомендовані товари; перехід у їх картки працює коректно
Кошик			
6.1	Додати товар у кошик із картки	P	Кнопка «Купити» в картці товару працює коректно – товар додається у кошик
6.2	Додати товар у кошик зі списку (якщо кнопка доступна)	P	Додавання товару зі списку категорії працює коректно; товар з’являється у кошику
6.3	Перевірити спливаюче вікно «товар додано»	P	Після додавання товару з’являється відповідне спливаюче вікно з підтвердженням
6.4	Перейти в кошик	P	Перехід у кошик працює коректно – сторінка кошика відкривається та відображає додані товари
6.5	Змінити кількість товарів	P	Зміна кількості товару працює коректно; інтерфейс оновлюється без затримок
6.6	Перевірити перерахунок загальної суми	P	Після зміни кількості товарів сума оновлюється коректно; арифметика знижок/бонусів обчислюється правильно
6.7	Видалити товар	P	Товар успішно видаляється з кошика; список товарів оновлюється
6.8	Очистити весь кошик	P	Після очищення всі товари видаляються з кошика
6.9	Перевірити порожній кошик	P	При порожньому кошику відображається відповідне повідомлення
Авторизація			
7.1	Відкрити форму авторизації	P	Форма входу відкривається після натискання іконки профілю

Id	Опис перевірки	Статус (P/F)	Коментар/Тестові дані
7.2	Виконати авторизацію з невалідним номером телефону	P	Тестові дані: +380000000000 Система відображає помилку «Невідомий код оператора». Авторизація не виконується
7.3	Виконати авторизацію з валідним номером телефону	P	Для валідного номера система показує вікно підтвердження СМС-коду
7.4	Виконати авторизацію через Google	P	Кнопка входу через Google доступна. Відкривається стандартне вікно OAuth Google
7.5	Виконати авторизацію через Apple	P	Кнопка входу через Apple доступна. Відкривається вікно авторизації Apple
7.6	Перевірити відображення даних користувача в кабінеті	P	Після входу відображаються персональні дані користувача: ім’я, номер телефону (логін), email, бонусний рахунок, списки товарів, адреси доставки, історія замовлень. Структура меню профілю завантажується коректно
7.7	Перевірити вихід з акаунта	P	Функція виходу доступна в меню профілю. Працює коректно
Оформлення замовлення			
8.1	Перейти до оформлення замовлення з кошика	P	Кнопка «Оформити замовлення» доступна та переводить на сторінку оформлення
8.2	Перевірити відображення списку товарів у замовленні	P	Усі додані в кошик товари правильно відображаються у блоці замовлення
8.3	Вибрати спосіб доставки	P	Доступні варіанти доставки відображаються
8.4	Вибрати спосіб оплати	P	Усі способі оплати доступні для вибору та працюють

Id	Опис перевірки	Статус (P/F)	Коментар/Тестові дані
8.5	Перевірити підсумкову вартість замовлення	P	Сума товарів, знижки та вартість доставки підсумовуються коректно
8.6	Перевірити можливість змінити контактні дані при оформленні	P	Поле контактного телефону доступне для редагування
8.7	Завершити оформлення (без фактичної оплати)	P	Замовлення успішно прийняте

За результатами проведеного тестування критичних функціональних дефектів не було виявлено. Усі основні користувацькі сценарії виконуються коректно, інтерфейс відображається стабільно, а бізнес-логіка працює відповідно до очікувань. Такий результат є типовим для зрілих проєктів, що тривалий час перебувають у продуктивній експлуатації та мають постійний технічний супровід і команду контролю якості.

3.3.2 Розробка тест-кейсів

Після виконання базового функціонального тестування за допомогою чек-листа виникла необхідність перевірити окремі функції веб-додатку більш детально. Для цього були розроблені тест-кейси, що дозволяють описати послідовність дій користувача, очікувані результати та зафіксувати фактичну поведінку системи.

Тест-кейси були створені та виконані у сервісі TestCaseLab, що забезпечило зручну структуру, можливість позначати статус виконання та робити коментарі до кожного кроку. У межах глибшого аналізу було підготовлено два тест-кейси: негативна перевірка валідації полів під час оформлення замовлення та комплексна перевірка роботи фільтрів і сортування

в категорії товарів. Отримані результати у вигляді скриншотів наведено нижче на рисунках 3.2 – 3.8.

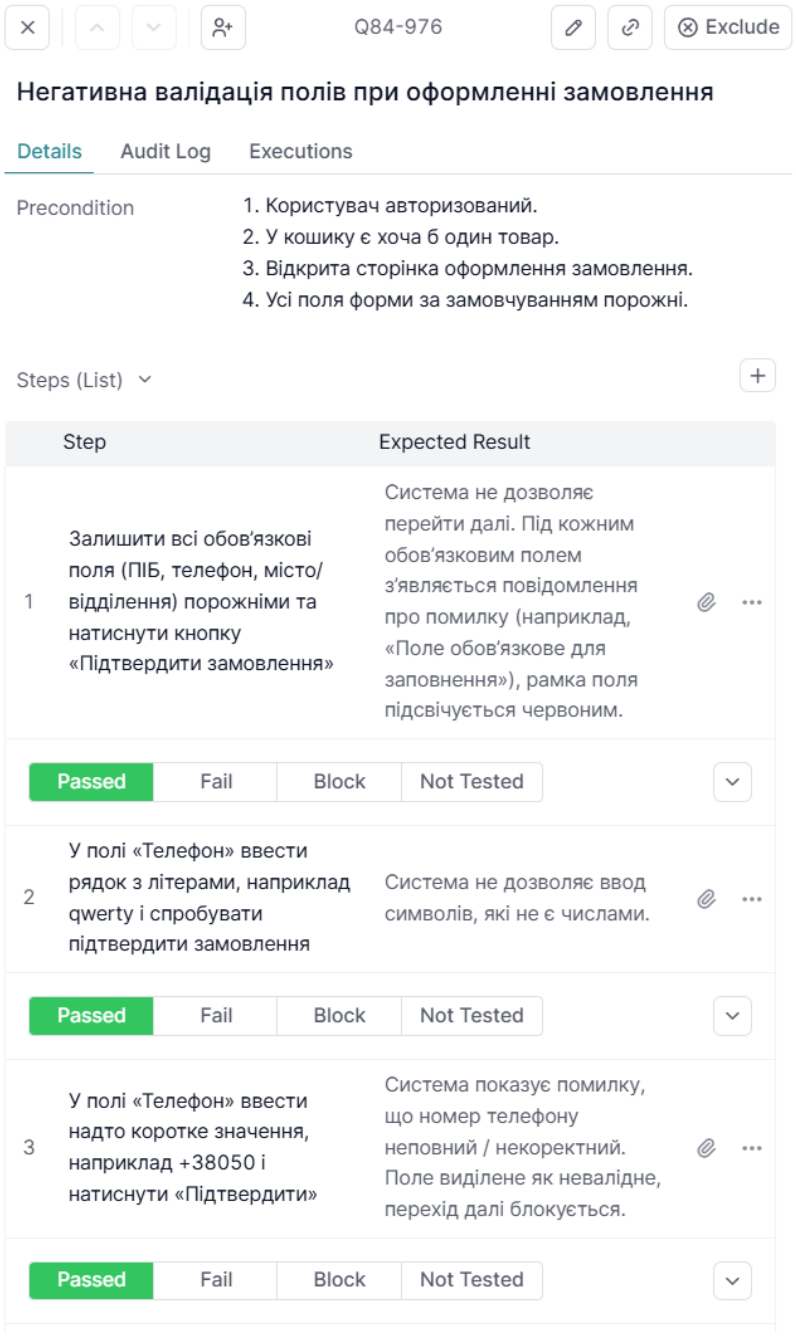


Рисунок 3.2 – Тест-кейс «Негативна валідація полів при оформленні замовлення» (частина 1)

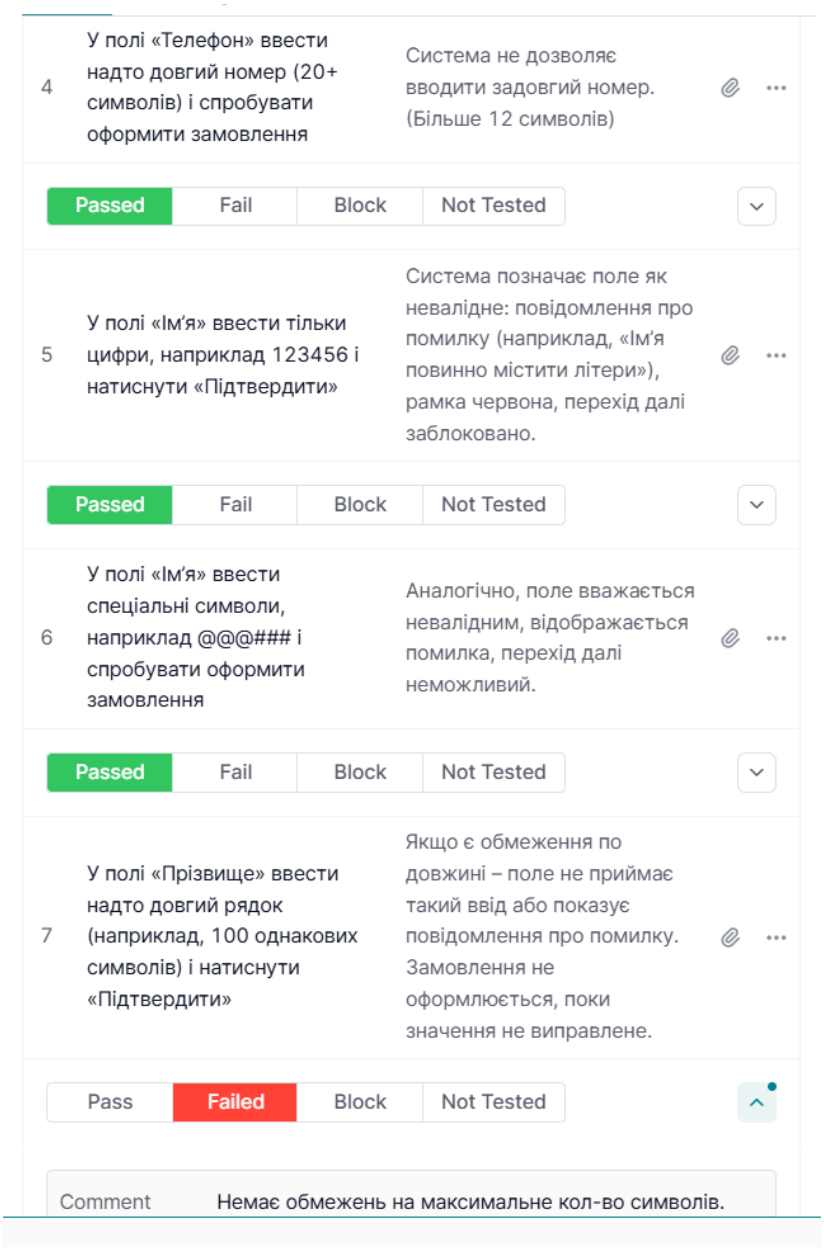


Рисунок 3.3 – Тест-кейс «Негативна валідація полів при оформленні замовлення» (частина 2)

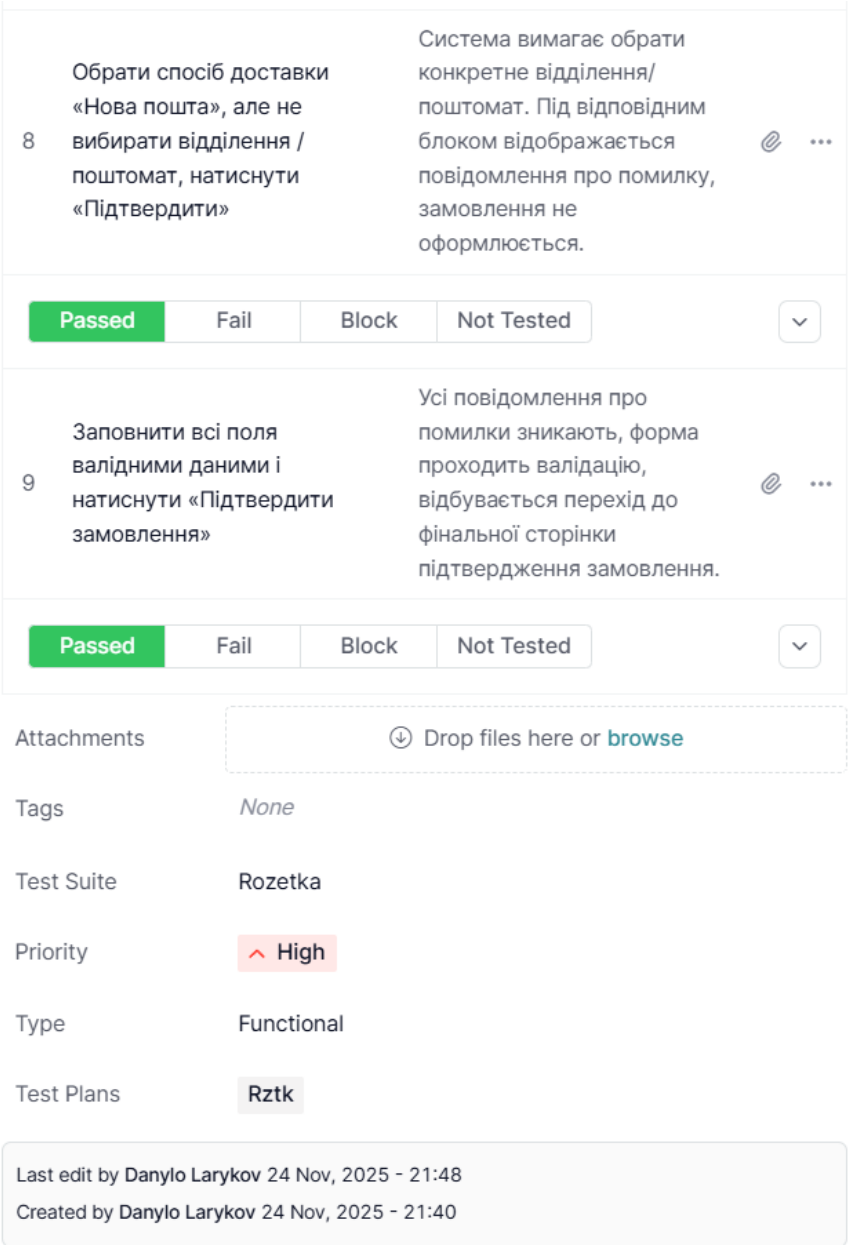


Рисунок 3.4 – Тест-кейс «Негативна валідація полів при оформленні замовлення» (частина 3)

У процесі виконання тест-кейсу було виявлено один дефект, пов’язаний з валідацією поля «Прізвище». Згідно з очікуваним результатом, система повинна запобігати введенню надто довгих значень та повідомляти користувача про помилку. Проте фактична поведінка відрізнялася: поле

приймає більше ніж 100 символів без будь-яких обмежень чи попереджень, а оформлення замовлення можливе навіть із некоректно довгим значенням.

Такий дефект може призвести до помилок під час обробки даних, некоректного відображення інформації в інтерфейсі або збоїв на стороні бекенду при збереженні нестандартно великих значень. Тому знайдену проблему можна класифікувати як валідаційний баг мінорного рівня. Фрагмент виконання тест-кейса та скриншот із фактичним результатом наведено нижче.

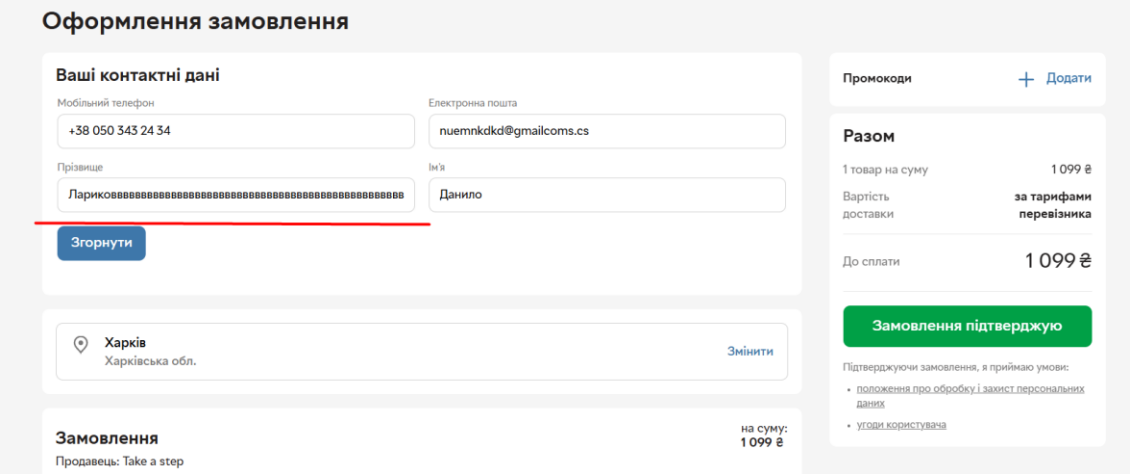


Рисунок 3.5 – Дефект валідації поля «Прізвище»

Другим розробленим тест-кейсом стала комплексна перевірка роботи фільтрів і сортування.

Метою даного тестування було оцінити коректність взаємодії декількох фільтрів одночасно, правильність оновлення списку товарів, збереження стану при переходах між сторінками та стабільність роботи механізмів сортування.

Усі кроки тест-кейсу були успішно виконані (статус Passed), збоїв у роботі фільтрів, втрати їхнього стану або порушення логіки сортування не виявлено.

Список товарів оновлювався відповідно до вибраних параметрів, активні фільтри зберігались при переходах між сторінками, а механізм сортування працював стабільно та передбачувано.

Скриншоти з TestCaseLab із зафіксованими результатами виконання наведено нижче.

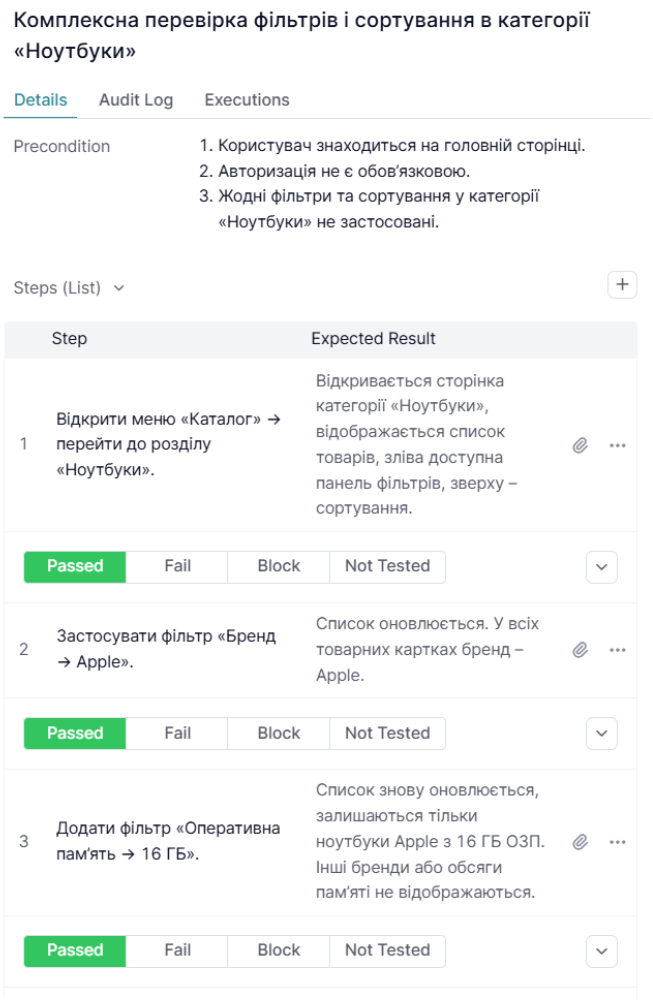


Рисунок 3.6 – Тест-кейс «Комплексна перевірка фільтрів і сортування» (частина 1)

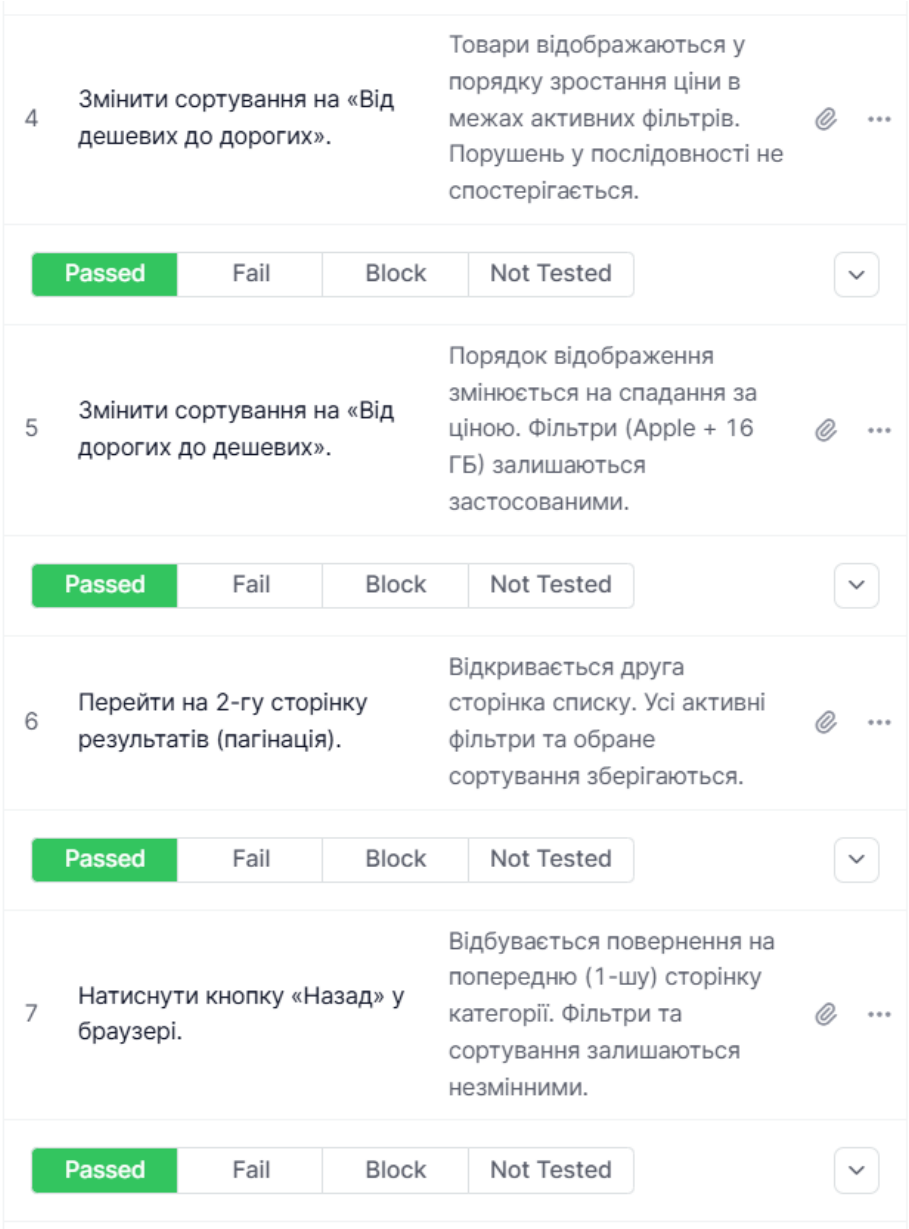


Рисунок 3.7 – Тест-кейс «Комплексна перевірка фільтрів і сортування» (частина 2)

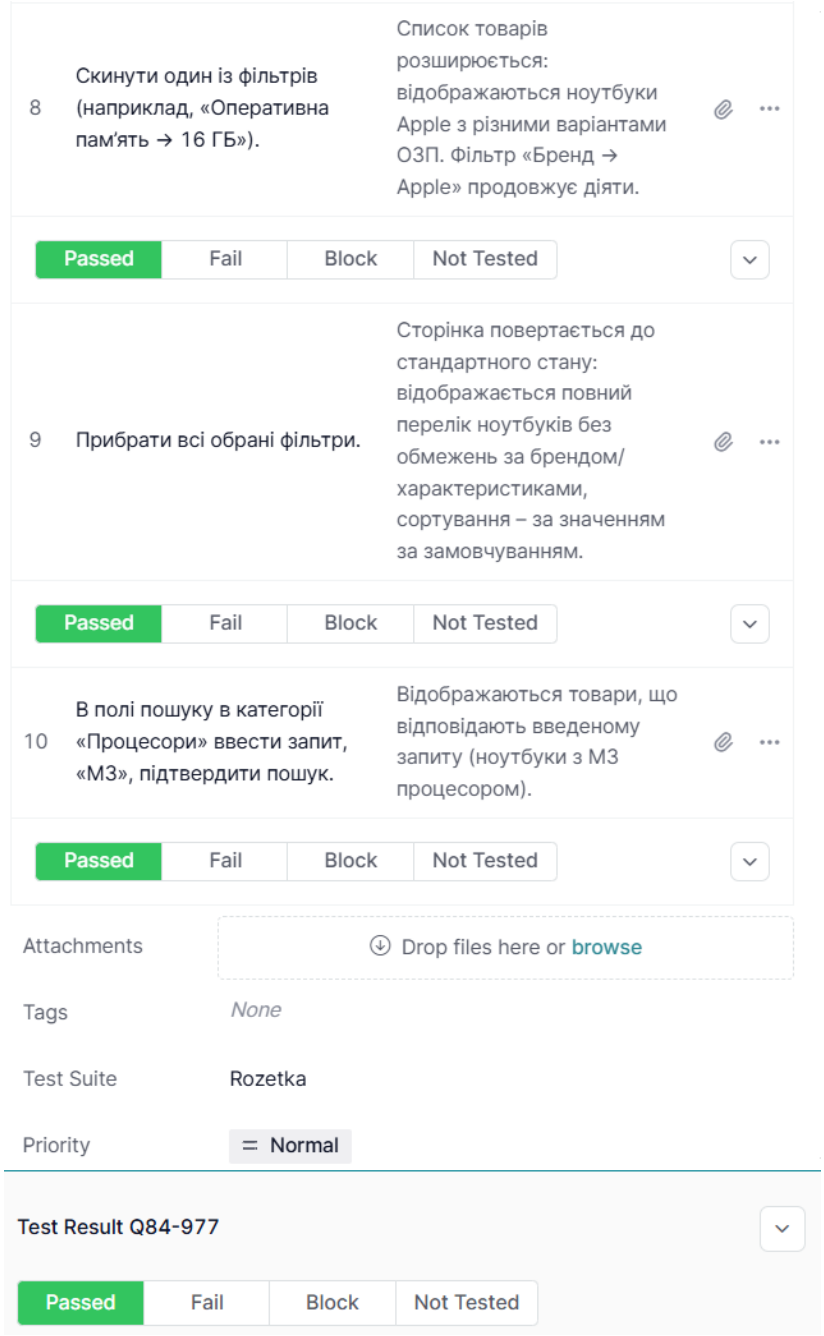


Рисунок 3.8 – Тест-кейс «Комплексна перевірка фільтрів і сортування» (частина 3)

У результаті виконання двох тест-кейсів було отримано повну картину щодо коректності роботи основних функціональних модулів веб-додатку

Rozetka.ua. Перший тест-кейс дозволив виявити дефект у валідації поля «Прізвище» під час оформлення замовлення, другий підтвердив стабільну роботу механізмів фільтрації та сортування. Отримані результати забезпечують необхідну основу для формування загальних висновків та подальшого аналізу якості веб-додатку.

3.4 Узагальнення результатів функціонального тестування

Проведене функціональне тестування охопило ключові користувацькі сценарії веб-додатку Rozetka.ua, включно з пошуком і фільтрацією товарів, навігацією, взаємодією з карткою товару, роботою кошика та оформленням замовлення. Усі базові перевірки за чек-листом пройшли успішно: сторінки завантажувалися стабільно, інтерфейс оновлювався коректно, а основна бізнес-логіка відпрацьовувала відповідно до очікуваного сценарію. Для глибшого аналізу були виконані два тест-кейси, спрямовані на перевірку валідації контактних даних та стійкості інтерфейсу при взаємодії з елементами управління. У ході перевірок було виявлено один дефект, пов’язаний із некоректною обробкою надмірно довгого значення у полі «Прізвище». Незважаючи на правильність обробки невалідних форматів (цифр, спецсимволів), система приймала надто довгі рядки без обмеження довжини та без позначення помилки, що може потенційно призводити до проблем під час збереження або відображення даних.

Узагальнюючи результати, можна стверджувати, що основний функціонал веб-додатку працює стабільно та відповідає очікуваній поведінці в більшості ключових сценаріїв. Виявлений дефект не блокує користувача і має локальний характер, однак його усунення покращить цілісність роботи форми та якість обробки персональних даних. Загалом проведене тестування підтверджує, що обрані методи дозволяють ефективно оцінити працездатність

веб-додатків класу e-commerce, своєчасно виявляти слабкі місця та формувати об’єктивне уявлення про якість продукту в умовах реального використання.

4 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ (ЧАСТИНА 2)

4.1 Загальні положення нефункціонального тестування

Нефункціональне тестування веб-додатків спрямоване на оцінювання характеристик, що визначають не логіку виконання функцій, а якість роботи системи загалом. На відміну від функціональних перевірок, які визначають коректність виконання конкретних сценаріїв, нефункціональні тести оцінюють такі властивості, як продуктивність, зручність використання, сумісність, безпека та стабільність роботи у різних умовах експлуатації.

Для e-commerce-платформи Rozetka.ua нефункціональні аспекти мають особливо високий пріоритет, адже веб-додаток обробляє великий обсяг трафіку, підтримує одночасні сесії тисяч користувачів, працює з великою кількістю інтерактивних елементів та виконує критично важливі бізнес-процеси – пошук товарів, застосування фільтрів, роботу кошика та оформлення замовлень.

Після проведення функціонального тестування та аналізу основних користувацьких сценаріїв було здійснено практичне дослідження нефункціональних характеристик веб-додатку Rozetka.ua.

Метою цього етапу є оцінювання продуктивності, сумісності, зручності використання та базових аспектів безпеки, що визначають стабільність роботи веб-додатку при різних умовах експлуатації та впливають на загальну якість користувацького досвіду.

4.2 Юзабіліті тестування

Юзабіліті-тестування веб-додатку Rozetka.ua проводилося з позиції кінцевого користувача. Оцінювання було спрямоване на визначення інтуїтивності інтерфейсу, логічності навігації, зручності виконання основних дій та загального комфорту взаємодії з системою [10].

Під час перевірки було проаналізовано ключові елементи інтерфейсу, що формують користувацький шлях.

1. Навігація та загальна структура інтерфейсу

Головні розділи розташовані логічно, головне меню добре помітне, а навігація між сторінками не викликає труднощів. Користувач легко знаходить потрібні розділи – каталог, кошик, авторизацію, фільтри та картки товарів.

2. Пошук і фільтрація товару

Поле пошуку має високу видимість та працює без затримок. Фільтри згруповані логічно, не приховані та одразу відображають доступні параметри. Система швидко оновлює результати, що підвищує зручність користування.

3. Інформативність картки товару

Картки товарів структуровані так, що користувач отримує всю необхідну інформацію без додаткових переходів: фото, характеристики, відгуки, наявність та рекомендації. Візуальні елементи не перевантажені, при цьому добре впорядковані.

4. Зрозумілість форм оформлення замовлення

Форми мають чіткі підказки, блоки згруповані логічно, порядок введення даних інтуїтивний. Під час тестування було виявлено один дефект: поле «Прізвище» дозволяє вводити надмірно довгі значення без обмеження довжини.

5. Видимість стану системи

Кнопки, повідомлення й завантаження сторінок реагують миттєво. Неможливо помилитися, чи була дія виконана – система оперативно змінює стан елементів (активні/неактивні кнопки, зміна кольору, повідомлення).

6. Узгодженість стилю

Веб-додаток використовує єдиний стиль на всіх сторінках: однакові кольори кнопок, поведінка елементів, структура блоків. Стиль витриманий, що зменшує когнітивне навантаження.

7. Візуальна та естетична складова

Інтерфейс стриманий, добре збалансований та не перевантажений. Акцент зроблено на ключових функціях: пошук, фільтри, категорії, картки товару та оформлення замовлення.

Оцінювання показало, що Rozetka.ua має зрозумілу структуру, передбачувану поведінку елементів та зручну логіку користування. Сайт легко освоюється, не створює когнітивного навантаження та підтримує швидке виконання всіх ключових дій – від пошуку товару до оформлення замовлення.

4.3 Тестування продуктивності

Тестування продуктивності використовувалося для оцінювання швидкості завантаження веб-додатку Rozetka.ua, стабільності інтерфейсу та ефективності виконання клієнтського коду. Оскільки це велика e-commerce платформа з великою кількістю динамічних елементів, якість роботи фронтенду має суттєвий вплив на користувацький досвід.

Для виконання перевірки використовувався інструмент Lighthouse, що входить до складу Google Chrome DevTools [4]. Щоб отримати коректні та

незалежні результати, тестування проводилося в режимі «Інкогніто», що дозволило усунути вплив локального кешу, авторизації та попередніх сесій. Усі розширення браузера були вимкнені, щоб виключити можливий вплив сторонніх скриптів на продуктивність. Режим тестування було встановлено як Navigation – Desktop, а для вимірювання обрано лише категорію Performance. Lighthouse автоматично застосовує обмеження продуктивності пристрою та мережі, тому результати не залежать від технічних характеристик комп’ютера. Для кожної сторінки тест повторювався декілька разів, після чого враховувалося середнє стабільне значення.

Для аналізу було відібрано три типові сторінки, що представляють різні рівні навантаження на інтерфейс:

- головна сторінка – містить банери, динамічний контент і велику кількість ресурсів;
- сторінка категорії «Ноутбуки» – характеризується великою кількістю зображень і фільтрів;
- картка товару – найбільш навантажена сторінка, що містить галереї, характеристики та рекомендації.

Результати наведено на рисунках 4.1–4.4.

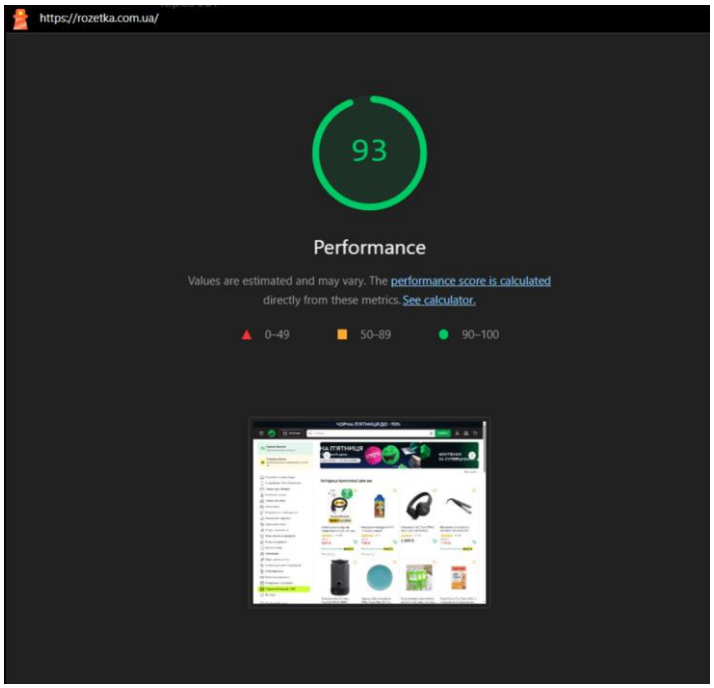


Рисунок 4.1 – Результати Perfomance testing для головної сторінки

Додатково для головної сторінки був використаний Lighthouse Scoring Calculator для перевірки структури зважування показників і підтвердження коректності обчислення підсумкового Performance Score [11].

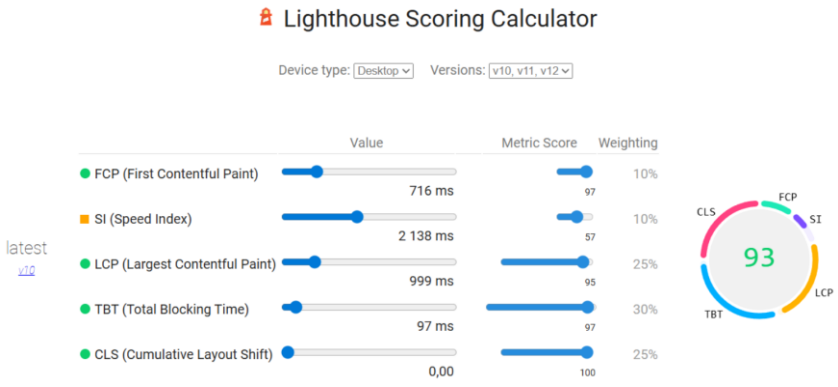


Рисунок 4.2 – Scoring Calculator Lighthouse для головної сторінки

Скріншоти Lighthouse для сторінки категорії «Ноутбуки» та картки товару «MacBook Air 13» додаються нижче.

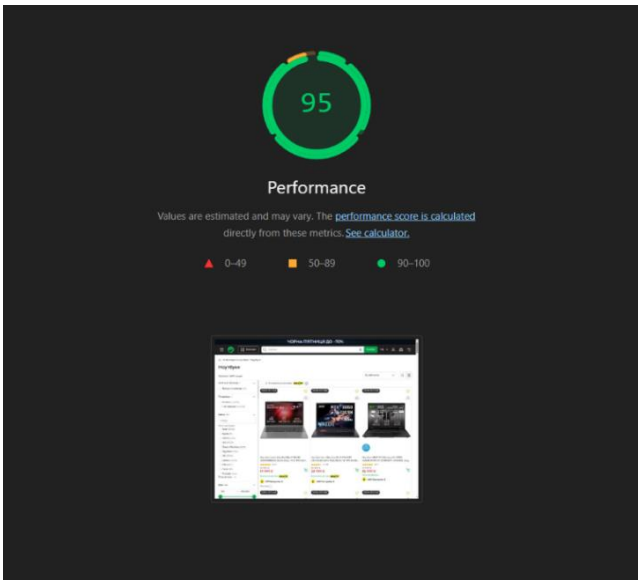


Рисунок 4.3 – Результати Perfomance testing для сторінки категорії «Ноутбуки»

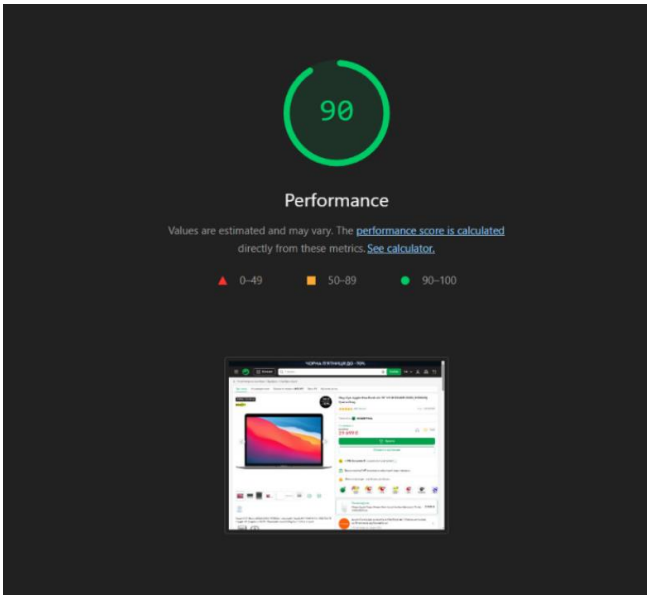


Рисунок 4.4 – Результати Perfomance testing для картки товару «MacBook Air 13»

Усі отримані числові показники заведено в таблицю.

Таблиця 4.1 – Результати тестування продуктивності

Сторінка	Performance Score	FCP	LCP	TBT	CLS	Speed index
Головна сторінка	93	0.7 с	1.0 с	97 мс	0.00	2.1 с
Категорія «Ноутбуки»	95	0.7 с	1.0 с	60 мс	0.000	1.8 с
Картка товару	90	0.8 с	1.0 с	140 мс	0.023	2.4 с

Отримані результати демонструють високий рівень оптимізації веб-додатку. Значення FCP та LCP у межах 0.7–1 секунди свідчать про швидке відображення основного контенту. Показники CLS майже нульові, що означає стабільність верстки без видимих зміщень. Час блокування основного потоку JavaScript залишається низьким (60–140 мс), тому інтерфейс не затримує взаємодію [5][14]. Найбільше навантаження спостерігається на картці товару через кількість зображень і динамічних блоків, однак навіть у цьому випадку сторінка зберігає високий рівень продуктивності.

4.4 Тестування кросбраузерної сумісності

Кросбраузерна перевірка була спрямована на визначення того, наскільки стабільно веб-додаток Rozetka.ua відтворює інтерфейс та обробляє дані в

різних браузерах. Для тестування були обрані три найпоширеніші браузери із актуальними версіями:

- Google Chrome v142
- Mozilla Firefox v142
- Microsoft Edge v145

Всі браузери були оновлені до актуальних стабільних версій перед початком перевірки. Тестування проводилося без авторизації, у режимі без кешу, з однаковою мовою та регіональними налаштуваннями, що дозволило звести до мінімуму вплив зовнішніх факторів. Масштабування інтерфейсу (zoom) у всіх браузерах було встановлено на 100%.

У процесі аналізу інтерфейсу (типографіка, структури макету, адаптивне розташування блоків, відображення карток товарів і фільтрів) значущих відмінностей не виявлено. Візуальна частина сайту відтворювалася однаково, що вказує на коректну реалізацію HTML/CSS і однакову роботу клієнтських скриптів у різних середовищах.

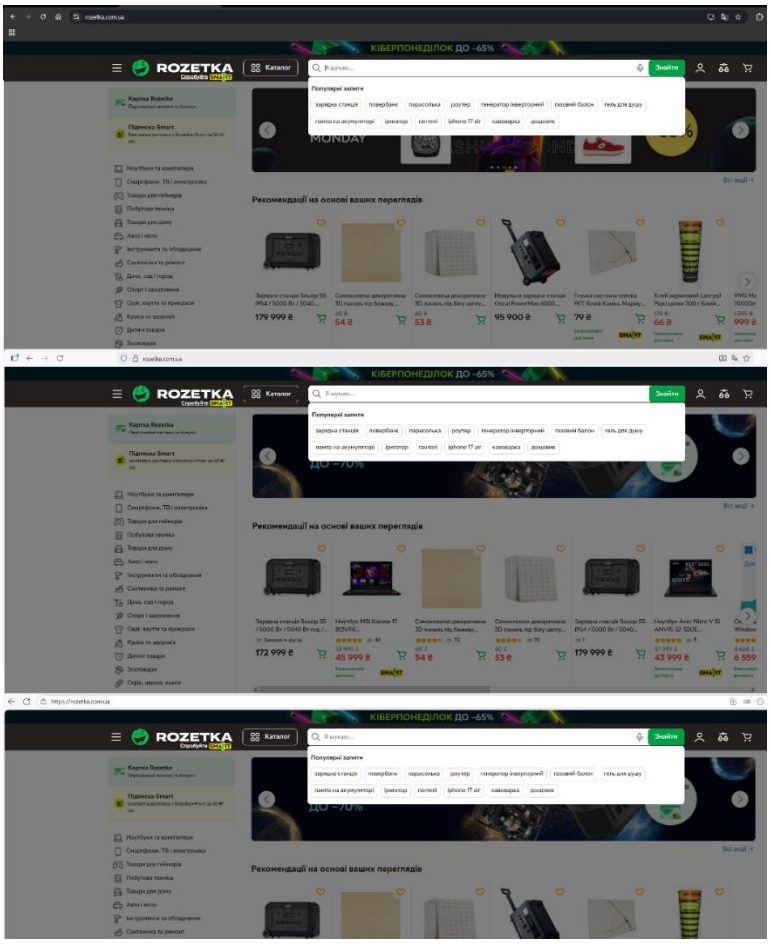


Рисунок 4.5 – Порівняння відображення інтерфейсу сторінки пошуку в Chrome, Firefox та Edge

Разом з тим, під час тестування було виявлено два ключові розходження у фактичній роботі веб-додатку.

Перше стосувалося різної кількості товарів, що відображалася при однаковому пошуковому запиті «зарядна станція» та ідентичних умовах.

Зафіксовані результати:

- Google Chrome - 784 товари
- Mozilla Firefox - 759 товарів
- Microsoft Edge - 742 товари

Розбіжність між мінімальним і максимальним значеннями сягала 42 позицій, що не може бути випадковістю або впливом кешу. Це може бути результатом різної обробки асинхронних запитів, різниць у сортуванні, специфіки CDN або A/B-тестування, яке розподіляє дані залежно від браузера або user-agent.

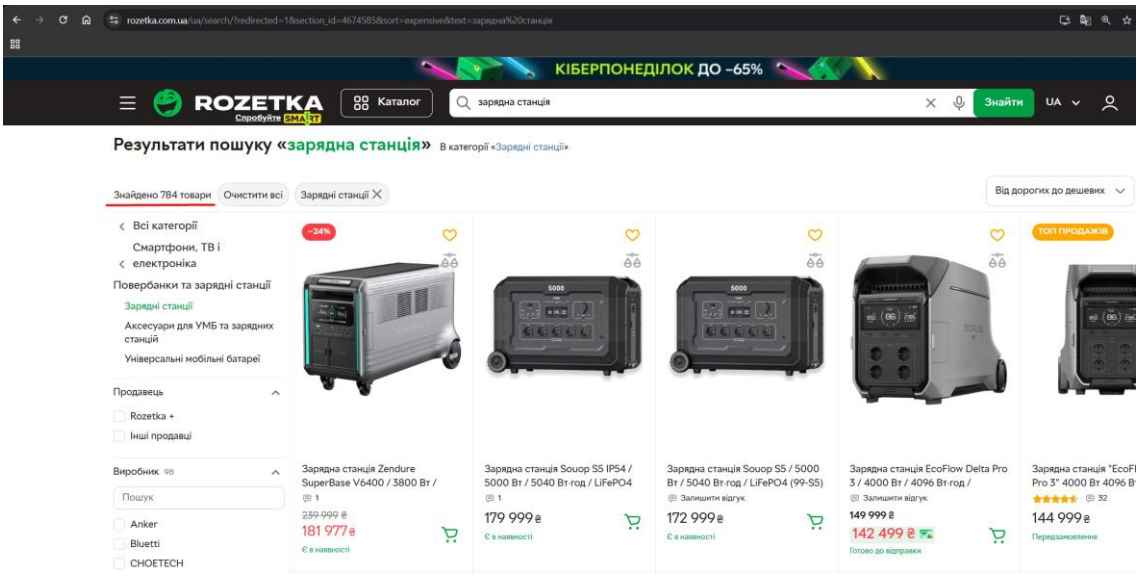


Рисунок 4.6 – Результати пошуку у Google Chrome

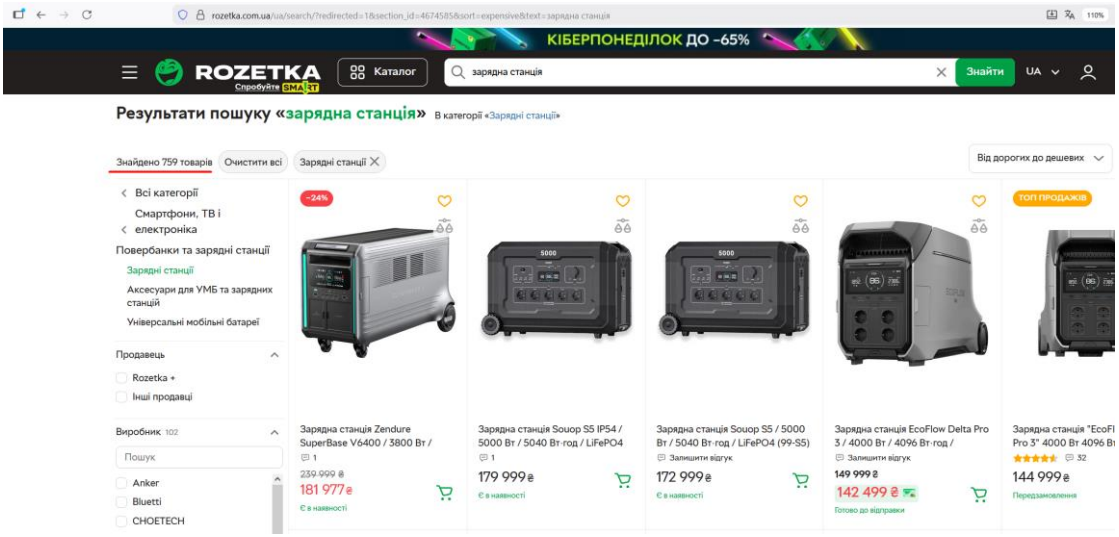


Рисунок 4.7 – Результати пошуку у Firefox

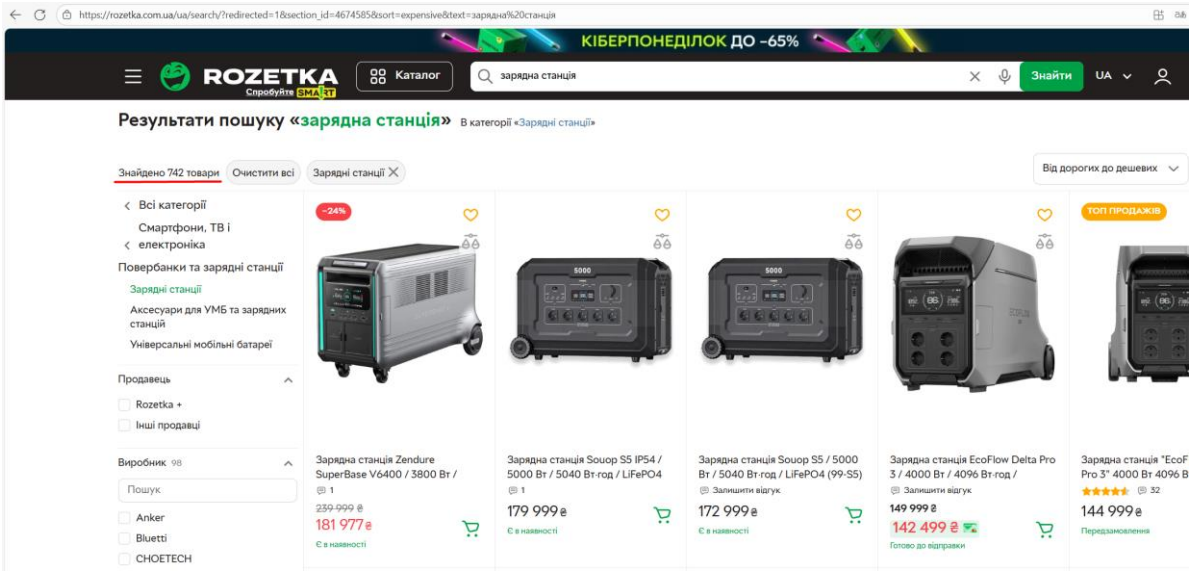


Рисунок 4.8 – Результати пошуку у Edge

Друга зафіксована відмінність стосувалася наявності додаткового функціонального елемента – кнопки запуску «Rozetka AI», яка з’являлася лише в браузері Microsoft Edge. В інших браузерах вона була відсутня. Це свідчить про вибіркову активацію функціоналу або використання експериментальних фіч для окремих user-agent’ів.

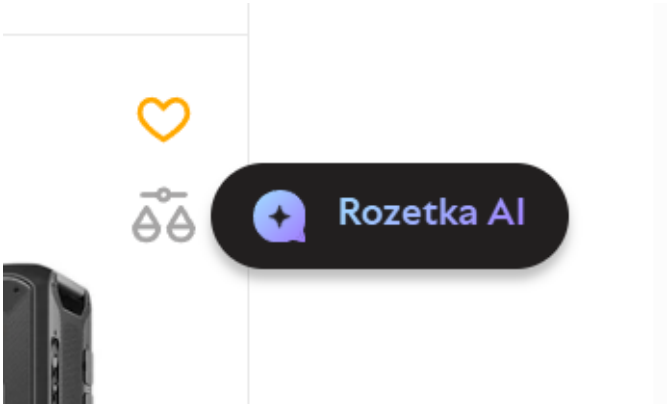


Рисунок 4.9 – Відображення кнопки «Rozetka AI» у Microsoft Edge

Загалом проведене тестування підтвердило, що інтерфейсна частина веб-додатку реалізована стабільно і не залежить від браузера. Водночас аналіз виявив різницю в кількості товарів при однакових умовах пошуку та додаткові функції, доступні лише в окремому браузері. Такі особливості підкреслюють важливість кросбраузерного тестування саме для динамічних веб-систем, де логіка роботи може відрізнятися залежно від середовища виконання.

4.5 Тестування безпеки

У межах практичної частини було виконано базове тестування безпеки веб-додатку Rozetka.ua. Мета цього етапу – оцінити найважливіші елементи захисту, які доступні для перевірки з боку клієнтської частини без втручання у внутрішню інфраструктуру сервісу.

Таке тестування є оглядовим (поверхневим), оскільки повноцінний аудит безпеки потребує доступу до серверної частини, вихідного коду, логів, а також дозволів на проведення інструментальних атак – що виходить за межі даної роботи та політик безпеки реального веб-додатку. Незважаючи на обмеження, проведений аналіз дозволяє оцінити базовий рівень захисту.

Аналіз виконувався за допомогою інструментів браузера Google Chrome DevTools [6], що дає змогу отримати достовірну інформацію щодо:

- параметрів HTTPS-з’єднання,
- політик безпеки у HTTP-заголовках,
- використання cookie-файлів,
- роботи сторонніх скриптів та загального стану клієнтської частини.

Для коректності результатів тестування перед його виконанням було вимкнено всі сторонні розширення, а перевірка проводилася у режимі «Інкогніто», що дозволяє мінімізувати вплив кешу та локальних даних.

1. Перевірка SSL-сертифіката та протоколу шифрування

Першим кроком було визначення стану HTTPS-з’єднання. Дані отримано у вкладці DevTools – Privacy and Security.

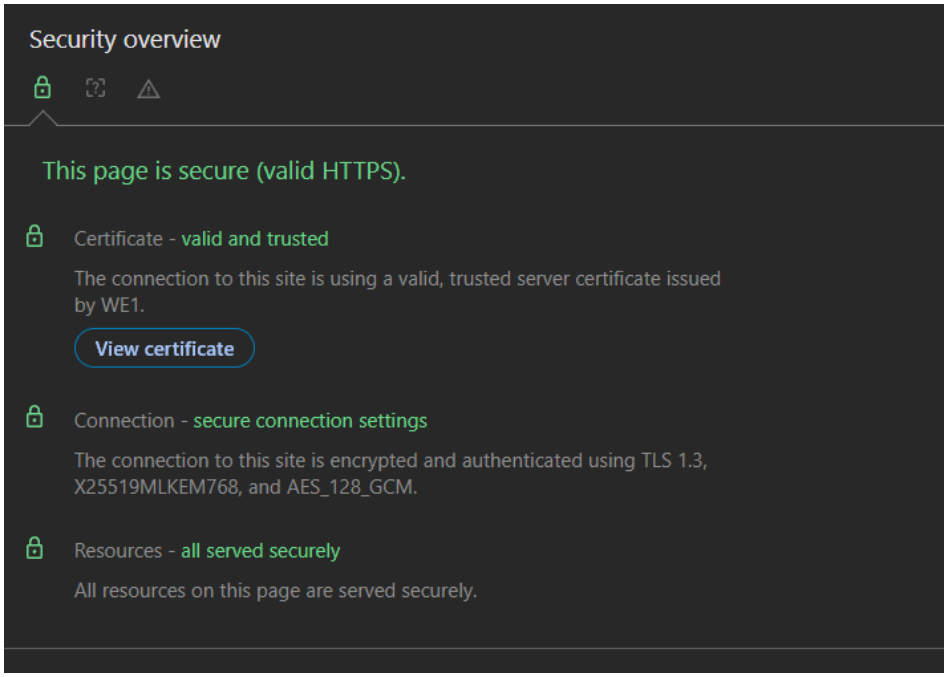


Рисунок 4.10 – Security overview

На сторінці тестування відображено:

- активне валідне HTTPS-з’єднання,
- чинний сертифікат, виданий центром сертифікації WE1,
- використання сучасного протоколу TLS 1.3,
- застосування криптографічних алгоритмів X25519 та AES-128-GCM.

Такий результат підтверджує, що передача даних між клієнтом і сервером відбувається у зашифрованому вигляді, а веб-додаток використовує сучасні механізми захисту.

2. Аналіз cookie-файлів та їхніх параметрів

Наступним етапом було вивчення cookie-файлів у розділі DevTools – Application – Cookies.

Name	Value	Domain	Path	Expires /...	Size	HttpOnly	Secure	SameSite	Partition...	Cross Site	Priority
__cf_bm	0GvHbCKBv0QRcU_NCQSPGuBpfmhSUgzNVyi362J.6hY...	.rozetka....	/	2025-12...	177	✓	✓	None			Medium
__exponea_etc__	d63dbda4-bc7a-447f-9673-243518c72892	.rozetka....	/	2027-01...	51		✓	None			Medium
__exponea_time2__	0.05258679389953613	.rozetka....	/	2025-12...	36		✓	None			Medium
__rtbh.lid	%7B%22eventType%22%3A%22lid%22%2C%22id%22...	rozetka....	/	2026-12...	139		✓	None			Medium
__utmz_gtm	utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)	.rozetka....	/	2026-06...	55						Medium
_clck	1cej520%5E2%5Eg1i%5E0%5E2162	.rozetka....	/	2026-12...	33						Medium
_clsk	1f92o27%5E1764679376807%5E4%5E0%5Es.clarity.ms...	.rozetka....	/	2025-12...	61						Medium
_fbp	fb.2.1764678420708.613018081812856355	.rozetka....	/	2026-03...	41			Lax			Medium
_ga	GA1.3.201884082.1764678420	.rozetka....	/	2027-01...	29						Medium
_ga_2WE9XBWEN9	GS2.3.s1764679245$o1$g0$t1764679375$j13$l0$h0	.rozetka....	/	2027-01...	59						Medium
_ga_3X15VBC9L9	GS2.3.s1764678420$o1$g1$t1764679378$j10$l0$h0	.rozetka....	/	2027-01...	59						Medium
_gcl_au	1.3.641600117.1764678420	.rozetka....	/	2026-03...	31						Medium
_gid	GA1.3.1025300182.1764678420	.rozetka....	/	2025-12...	31						Medium
_tgidts	eyJzaCI6ImQ0MWQ4Y2Q5OGYwMGIyMDRlOTgwMDk5...	.rozetka....	/	2025-12...	179		✓	None			Medium
_tglksd	eyJzIjoiZWQ4MDdjZTktODQ4Yi00YzM5LWJhYTUtMGZ...	.rozetka....	/	2026-12...	187		✓	None			Medium
_tgpc	1d474aea-8971-4197-b206-a33cc13d614b	.rozetka....	/	2026-12...	41		✓	None			Medium
_tgsid	eyJscGQiOiJ7XCJscHVcljpclmh0dHBzOi8vcm96ZXRrYS5...	.rozetka....	/	2025-12...	646		✓	None			Medium
_tguatd	eyJzYyI6IihkaXJIY3QpIiwiZnRzIjoiKGRpcmVjdCkifQ==	.rozetka....	/	2025-12...	55		✓	None			Medium
_tt_enable_cookie	1	.rozetka....	/	2026-12...	18						Medium
_ttp	01KBFGFK2C2YQXBJZ5DJV1EHSQ_.tt.2	.rozetka....	/	2026-12...	36						Medium

Рисунок 4.11 – Перелік cookies

Після перевірки cookie-файлів, можна сказати, що більшість з них мають прапор Secure, котрий блокує їх передачу незашифрованими каналами. Критично важливі cookie мають атрибут HttpOnly, що запобігає доступу до них через JavaScript, а частина cookie використовує політику SameSite=Lax, однак деякі сторонні cookie мають значення None, що типово для інтеграції зі сторонніми сервісами.

3. Перевірка заголовків безпеки HTTP

У вкладці DevTools – Network проаналізовано відповіді сервера.

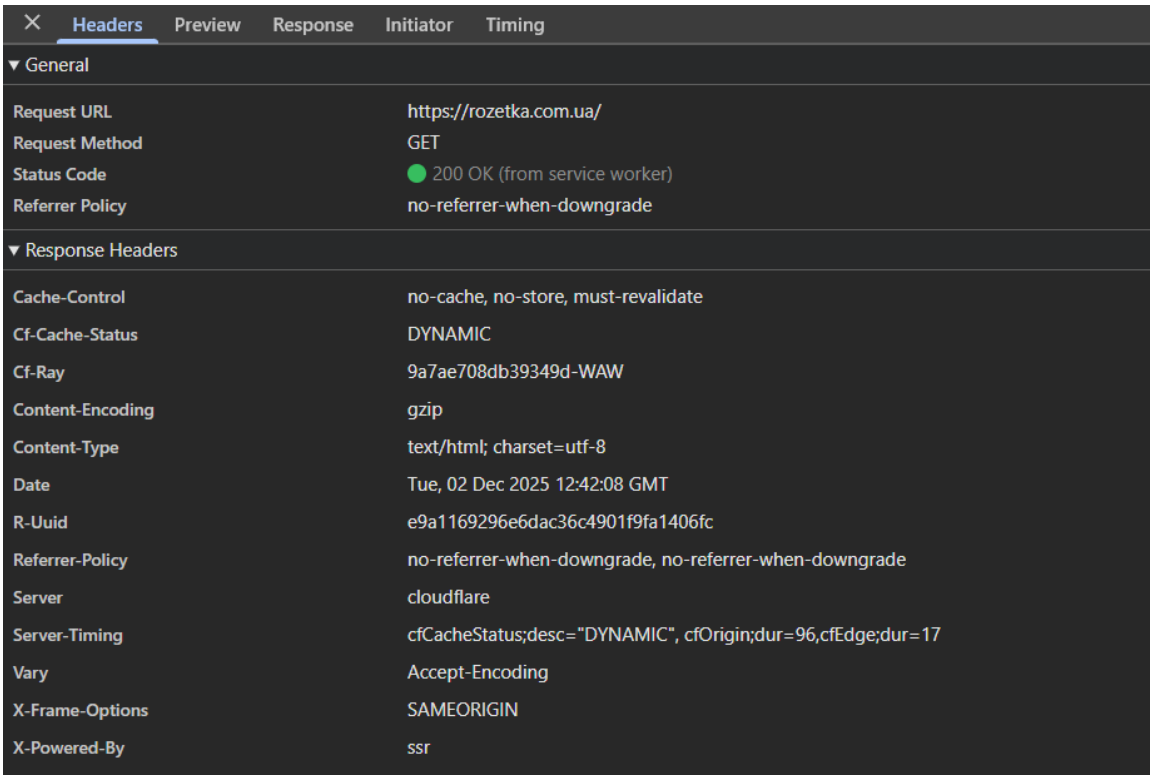


Рисунок 4.12 – Заголовки відповіді для GET запиту

Після аналізу було виявлено наступні важливі заголовки:

- Cache-Control: no-cache, no-store, must-revalidate дозволяє уникнути кешування HTML-контенту, що покращує безпеку;
- X-Frame-Options: SAMEORIGIN – запобігає атакам типу Clickjacking;
- Content-Type: text/html; charset=utf-8 – коректно визначений;
- Server: cloudflare – використовується CDN та захист від DDoS;
- Referrer-Policy: no-referrer-when-downgrade – помірний рівень захисту, без жорсткої політики confidentiality.

Результат свідчить про наявність базового набору механізмів безпеки, які відповідають сучасним вимогам до веб-застосунків [3][12].

Підсумовуючи, можна зазначити, що веб-додаток Rozetka.ua демонструє прийнятний рівень базового клієнтського захисту. Більш глибоке тестування неможливе в межах роботи, оскільки потребує доступу до внутрішніх

компонентів системи й виконання дій, що виходять за рамки етичного та дозволеного дослідження.

4.6 Узагальнення результатів нефункціонального тестування

У четвертому розділі було досліджено ключові нефункціональні характеристики веб-додатку Rozetka.ua, що дозволило визначити його якість у реальних умовах використання. Проведене юзабіліті-тестування показало, що інтерфейс є логічним, послідовним та зручним для користувача: навігація інтуїтивна, структура сторінок зрозуміла, а основні дії виконуються без зайвого когнітивного навантаження. Суттєвих UX-проблем під час оцінювання не виявлено.

Перевірка продуктивності підтвердила високу швидкість завантаження сторінок, стабільність роботи інтерфейсу та оптимізованість клієнтської частини: усі протестовані сторінки продемонстрували високі значення Performance Score та прийнятні показники ключових метрик. Кросбраузерне тестування засвідчило коректність відображення інтерфейсу у різних браузерах, хоча виявило відмінності у кількості товарів за однаковим пошуковим запитом та появу додаткового функціонального елемента лише в Edge, що підкреслює специфіку обробки динамічного контенту в різних середовищах. Базова перевірка безпеки підтвердила коректну роботу механізмів захисту на клієнтському рівні, зокрема налаштувань HTTPS, cookie-параметрів та заголовків безпеки.

Загалом результати нефункціонального тестування демонструють, що веб-додаток Rozetka.ua відповідає очікуванням для сучасних e-commerce систем: працює стабільно, швидко, без помітних збоїв у відображенні інтерфейсу та забезпечує достатній рівень захисту користувацьких даних.

Отримані спостереження доповнюють результати функціонального тестування та формують цілісне уявлення про якість веб-додатку.

ВИСНОВКИ

У кваліфікаційній роботі здійснено комплексне дослідження методів тестування веб-додатків та їх практичного застосування на прикладі високонавантаженого e-commerce ресурсу Rozetka.ua. Проведений аналіз дозволив сформувати цілісне уявлення про особливості функціонального й нефункціонального тестування сучасних веб-систем.

У ході роботи було:

- систематизовано класифікацію веб-додатків та методів їх тестування;
- проведено аналіз ключових сценаріїв користувача та виконано функціональні перевірки веб-додатку;
- досліджено юзабіліті інтерфейсу з позиції кінцевого користувача;
- виконано тестування продуктивності за допомогою інструмента Lighthouse;
- проведено кросбраузерну перевірку роботи веб-додатку у трьох популярних браузерах;
- здійснено базові клієнтські перевірки безпеки, включаючи аналіз HTTPS-з’єднання, параметрів cookie та заголовків HTTP.

Результати практичного дослідження засвідчили, що веб-додаток Rozetka.ua демонструє високий рівень оптимізації, стабільність інтерфейсу та якісну реалізацію більшості користувацьких сценаріїв. Показники продуктивності знаходяться на високому рівні, а структура інтерфейсу забезпечує інтуїтивну та швидку взаємодію користувача з системою. Водночас було виявлено окремі особливості, такі як неконсистентність результатів пошуку у різних браузерах та поява експериментальних функцій лише для окремих user-agent’ів.

Проведене тестування дозволило визначити важливі аспекти забезпечення якості веб-додатків і підтвердило доцільність застосування комплексного підходу, що охоплює функціональні та нефункціональні методи оцінювання. Отримані результати можуть бути використані під час подальшого вдосконалення веб-систем, а також слугувати основою для розширення практик тестування у реальних проєктах.

Таким чином, поставлені завдання дослідження виконано у повному обсязі, а мета роботи досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ISO/IEC 25010:2016. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). URL: https://www.iso.org/standard/35733.html (дата звернення: 02.10.2025)
2. ISTQB. Glossary of Software Testing Terms. URL: https://glossary.istqb.org (дата звернення: 02.10.2025)
3. OWASP Foundation. OWASP Top 10 – Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 28.10.2025)
4. Google Developers. Lighthouse Documentation. URL: https://developer.chrome.com/docs/lighthouse/ (дата звернення: 20.11.2025)
5. Google Developers. Web Vitals Documentation. URL: https://web.dev/vitals/ (дата звернення: 23.11.2025)
6. Chrome Developers. Chrome DevTools Documentation. URL: https://developer.chrome.com/docs/devtools/ (дата звернення: 23.11.2025)
7. Mozilla Foundation. MDN Web Docs – Web Applications Overview. URL: https://developer.mozilla.org/ (дата звернення: 28.07.2025)
8. Intelivita. Types of Web Applications: Complete Overview. URL: https://www.intelivita.com/blog/types-of-web-applications/ (дата звернення: 22.10.2025)

9. Webcase Studio. Що таке веб-додаток: класифікація та особливості. URL: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 24.10.2025)
10. Nielsen Norman Group. 10 Usability Heuristics for User Interface Design. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 15.11.2025)
11. Google Chrome. Lighthouse Scoring Calculator. URL: https://googlechrome.github.io/lighthouse/scorecalc/ (дата звернення: 28.11.2025)
12. OWASP Cheat Sheet Series. Input Validation, Authentication, Secure Coding Practices. URL: https://cheatsheetseries.owasp.org/ (дата звернення: 30.11.2025)
13. W3C. Web Architecture & Web Standards. URL: https://www.w3.org/ (дата звернення: 10.11.2025)
14. Google Developers. PageSpeed Insights. URL: https://developers.google.com/speed/docs/insights/v5/about? (дата звернення: 1.12.2025)
15. Лариков Д., Захватова Т., Єгоров А. Методи тестування програмного забезпечення // Research in Science: Technology and Economics : матеріали 5-ї Міжнар. наук.-практ. конф. (10–12 грудня 2025 р., Люксембург). Luxembourg : International Scientific Unity, 2025. С. 263–266. DOI: https://doi.org/10.70286/isu-10.12.2025