МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
_____ д. е. н., проф. П. М. Грицюк
«_____» _____2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня **«**бакалавр**»**
за освітньо-професійною програмою «Інформаційні системи та технології»
спеціальності 126 «Інформаційні системи та технології»

на тему: «Інформаційна система e-commerce-платформи: методи та засоби тестування вебзастосунків з реалізації домашнього текстилю на базі CMS OpenCart»

Виконав:
здобувач вищої освіти 4 курсу, групи ІСТ-41
Ружицький Владислав Романович

Керівник:
к.т.н., доцент Джоші О. І.

Рецензент:
д.е.н., професор Грицюк П.М.

Рівне – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра комп'ютерних технологій та економічної кібернетики

Освітньо-кваліфікаційний рівень – бакалавр
Освітньо-професійна програма «Інформаційні системи і технології»
Спеціальність 126 «Інформаційні системи і технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д.е.н., професор П.М. Грицюк

_____
"____" _____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачу ***Ружицькому Владиславу Романовичу***
(прізвище, ім'я, по батькові)

1. Тема роботи ***Інформаційна система e-commerce-платформи: методи та засоби тестування вебзастосунків з реалізації домашнього текстилю на базі CMS OpenCart***

керівник роботи: *Джоші Олена Іванівна, к.т.н.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від *«23» квітня 2026* р. *С №-476*

2. Термін здачі студентом закінченої роботи *15.06.2026 р.*

3. Вихідні дані до роботи: *1) документація щодо e-commerce-платформи fionahome.com.ua; 2) Інтернет-ресурси щодо теми кваліфікаційної роботи; 2) наукові публікації за темою кваліфікаційної роботи.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Теоретичні дослідження (аналіз предметної області; систематизація методів тестування вебзастосунків, обґрунтування вибору інструментального стеку); 2) розробка концептуальної моделі ІС; 3) програмна реалізація backend-у та frontend-у ІС; 4) реалізація модуля автоматизованого тестування; 5) аналіз результатів тестування*

5. Перелік графічного матеріалу *1) обґрунтування вибору інструментального стеку; 2) ER-модель ІС; 3) інтерфейс; 4) результати тестування та їх аналіз*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийнв
1	*Джоші Олена Іванівна, доцент*		
2	*Джоші Олена Іванівна, доцент*		
3	*Джоші Олена Іванівна, доцент*		

Дата видачі завдання *«19» січня 2026* р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів (роботи)	Примітка
1	Огляд літератури, збір статистичних даних	01.02.2026–15.03.2026	
2	Розділ 1.	15.03.2026–15.04.2026	
3	Розділ 2.	15.04.2026–01.05.2026	
4	Розділ 3.	01.05.2026–15.05.2026	
5	Підготовка пояснювальної записки	15.05.2026–25.05.2026	
6	Написання вступу, висновків, реферату	25.05.2026–01.06.2026	
7	Підготовка презентації роботи	01.06.2026–09.06.2026	
8	Попередній захист роботи	09.06.2026	
9	Відгук керівника, рецензування роботи, перевірка на плагіат	09.06.2026–15.06.2026	
10	Допуск до захисту	15.06.2026	

Студент _____Ружицький В. Р.
(підпис) (прізвище і ініціали)

Керівник кваліфікаційної роботи

к.т.н., доцент _____Джоші О.І.
(підпис) (прізвище і ініціали)

ЗАЯВА

щодо самостійності виконання випускної кваліфікаційної роботи

Я, _____(ПІБ), студент(ка) _____курсу ___ групи ___ ННІ _____ заявляю: моя випускна кваліфікаційна робота на тему «_____

_____

_____» (назва роботи), яка надається в екзаменаційну комісію із захисту бакалаврської роботи _____

_____(назва спеціальності) для захисту, виконана самостійно і не містить плагіату. Всі запозичення з друкованих та електронних джерел, у тому числі із захищених раніше випускових кваліфікаційних робіт, кандидатських і докторських дисертацій мають відповідні посилання. Я не використовував(ла) шахрайські методи заміни символів чи інші способи модифікації тексту. Я ознайомлений(а) з чинним Порядком перевірки навчальних, кваліфікаційних, навчально-методичних та наукових робіт на наявність ознак академічного плагіату в НУВГП та Положенням про академічну доброчесність в НУВГП, за яким виявлення плагіату є підставою для відмови в допуску моєї роботи до захисту та застосування відповідних санкцій (академічної відповідальності).

StrikePlagiarism.com | Національний університет водного господарства та природокористування

Документ прийнятий

Метадані

ДОКУМЕНТ

Заголовок
2026_Ruzhytskyi_126_bakalavr_.docx

Автор	Науковий керівник / Експерт	ІД документу
Ружицький Владислав Романович	**Ружицький Владислав Романович**	**334310280**

ОРГАНІЗАЦІЯ

Назва організації	підрозділ
National University of Water and Environmental Engineering	**National University of Water and Environmental Engineering**

ЗВІТ

Дата звіту	Дата редагування
6/14/2026	**---**

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

6.04% (6.04%) КП 1	3.98% (3.98%) КЦ
14461 Кількість слів	**108115** Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		1
Інтервали		1
Мікропробіли		58
Білі знаки		0
Парафрази (SmartMarks)	a	53

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ep3.nuwm.edu.ua/19464/1/04-05-49%D0%9C.pdf	49 (0.34 %)

А К Т
перевірки випускової кваліфікаційної роботи
автора Ружицького Владислава Романовича
на рівень запозичень та можливих маніпуляцій з текстом

Відповідно до даних сервісу StrikePlagiarism файл «**2026_Ruzhytskyi_126_bakalavr.docx**», містить: ___ % коефіцієнта подібності; _____ % коефіцієнта цитованості; _____ замінених букв; ___ інтервалів; ___ мікропробілів; ____ білих знаків; _____ парафрази.

За результатами перевірки засвідчую, що:
Автор кваліфікаційної роботи **Ружицький Владислав Романович:**

- самостійно виконав (ла) кваліфікаційну роботу;
- коректно посилався (лась) на використані інформаційні джерела;
- в роботі відсутні ознаки академічної недоброчесності.

Керівник кваліфікаційної роботи _____ (О.І.Джоші)
(підпис)

_____
(дата)

АНОТАЦІЯ

Ружицький В. Р. Інформаційна система e-commerce-платформи: методи та засоби тестування вебзастосунків з реалізації домашнього текстилю на базі CMS OpenCart. Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»: 85 ст., 22 рис., 25 табл., 3 додатки на 16 ст., 50 літературних джерел.

Об'єкт дослідження – інформаційна система e-commerce-платформи.

Предмет дослідження – методи та засоби тестування вебзастосунків.

Методи дослідження – методи аналізу джерел інформації, системного аналізу, класифікації, моделювання, системного підходу, а також програмування та тестування програмної реалізації та ефективності системи.

Кваліфікаційна робота присвячена комплексному тестуванню e-commerce-платформи для продажу домашнього текстилю на базі CMS OpenCart. У роботі проведено аналіз предметної області, систематизовано методи тестування вебзастосунків, обґрунтовано вибір інструментального стеку. Проведено аналіз ключових SQL-запитів, реалізовано модуль автоматизованого E2E-тестування за патерном Page Object Model. Виконано тестування за сімома напрямами, виявлено та задокументовано дефекти, сформульовано рекомендації щодо підвищення якості платформи.

Ключові слова: e-commerce, OpenCart, тестування вебзастосунків, Selenium WebDriver, Page Object Model, SQL-запити, адаптивний frontend, продуктивність, API-тестування, крос-браузерність.

ЗМІСТ

ВСТУП

Розвиток цифрової економіки в Україні супроводжується стрімким зростанням ринку електронної комерції, що в умовах воєнного часу набуло особливого прискорення через масовий перехід споживачів і підприємців до онлайн-каналів торгівлі. Інтернет-магазини стали основним або єдиним каналом збуту для значної частини підприємств малого і середнього бізнесу, що зумовлює підвищені вимоги до якості, стабільності і безпеки відповідних вебзастосунків. У цьому контексті забезпечення якості програмного забезпечення e-commerce-платформ набуває не лише технічного, але і прямого комерційного значення: будь-який збій у роботі платіжного шлюзу, некоректна поведінка кошика або недоступність сторінки оформлення замовлення призводять до безпосередніх фінансових втрат і підривають довіру покупців.

Незважаючи на широке розповсюдження готових CMS-рішень для побудови інтернет-магазинів, питання систематичного тестування вебзастосунків на їх основі залишається практично невирішеним для переважної більшості малих і середніх e-commerce-підприємств в Україні. Більшість із них не має виділеної QA-команди і не застосовує жодних формалізованих методів тестування, обмежуючись ситуативною ручною перевіркою після внесення змін. Така ситуація створює системний ризик появи дефектів у виробничому середовищі, що безпосередньо впливає на рівень конверсії і задоволеність покупців.

Метою кваліфікаційної роботи є дослідження і практична реалізація комплексу методів та інструментів тестування вебзастосунків стосовно функціонуючої e-commerce-платформи домашнього текстилю на базі CMS OpenCart, що забезпечують систематичне виявлення дефектів і підтвердження якості системи.

Досягнення поставленої мети передбачає вирішення таких **завдань**:

1. Аналіз предметної області e-commerce-платформ для домашнього текстилю та особливостей CMS OpenCart як технологічної основи системи.
2. Систематизація методів тестування вебзастосунків і обґрунтування вибору інструментального стеку, що відповідає технологічним особливостям досліджуваної платформи.
3. Розробка концептуальної моделі інформаційної системи, ER-модель бази даних і стратегії тестування платформи.
4. Програмна реалізація backend і frontend частин платформи, аналіз ключових SQL-запитів системи.
5. Реалізація модуля автоматизованого тестування на основі Selenium WebDriver і pytest за патерном Page Object Model та проведення тестування критичних сценаріїв платформи.
6. Аналіз результатів тестування, документування виявлених дефектів і формулювання рекомендації щодо підвищення якості та продуктивності платформи.

Об'єктом дослідження є інформаційна система e-commerce-платформи.

Предметом дослідження є методи та засоби тестування вебзастосунків.

У процесі виконання дослідження використовувалися такі методи: системного аналізу – для дослідження архітектури e-commerce-платформи і виявлення її ключових компонентів; функціонального моделювання – для побудови концептуальної моделі бізнес-процесів і діаграм варіантів використання; реляційного моделювання – для побудови ER-моделі бази даних і аналізу її нормалізації; тестування на основі ризиків – для пріоритизації тест-кейсів відповідно до впливу потенційних дефектів на бізнес; функціонального і регресійного тестування чорної та сірої скриньки – для перевірки коректності реалізації бізнес-логіки; методи навантажувального тестування і аудиту продуктивності – для оцінки поведінки системи при різних умовах навантаження; автоматизованого тестування із застосуванням патерну

Page Object Model – для побудови стабільного і підтримуваного набору автоматизованих тест-кейсів.

Наукова новизна роботи полягає у розробці комплексної стратегії тестування, адаптованої до специфіки CMS OpenCart і предметної області домашнього текстилю, та у практичній реалізації модуля автоматизованого тестування за патерном Page Object Model, що охоплює всі критичні сценарії e-commerce-платформи від перегляду каталогу до підтвердження замовлення.

Практичні результати отримано шляхом безпосереднього дослідження функціонуючого комерційного вебзастосунку fionahome.com.ua – інтернет-магазину домашнього текстилю, що спеціалізується на реалізації постільної білизни, рушників, покривал та супутньої продукції. Платформа підтримує онлайн-оплату банківськими картками Visa та Mastercard, а також оплату при отриманні, і інтегрована з API служби доставки Нова Пошта..

Практична значущість роботи визначається тим, що розроблений модуль автоматизованого тестування може бути безпосередньо впроваджений у процес супроводу платформи fionahome.com.ua, а спроектована стратегія тестування є масштабованою і придатною для застосування на інших e-commerce-платформах аналогічного класу на базі OpenCart. Виявлені в ході тестування дефекти і сформульовані рекомендації щодо оптимізації продуктивності мають конкретний прикладний характер і можуть бути реалізовані у поточному циклі супроводу системи.

Матеріали роботи апробовані в умовах реального функціонуючого підприємства. Аудит продуктивності засобами Google Lighthouse і аналіз SQL-запитів за допомогою команди EXPLAIN надали конкретні вимірювані дані про поточний стан системи, придатні для порівняння після впровадження рекомендованих оптимізацій.

Програмні продукти, що використовувались під час написання кваліфікаційної роботи: Python, IDE PyCharm, VS Code, Google Chrome.

Під час підготовки цього наукового дослідження використовувалися інструменти штучного інтелекту: ChatGPT. Вони застосовувалися для пошуку літератури, стилістичної корекції. Усі розділи роботи, під час написання яких використовувався ШІ, були перевірені та відредаговані автором особисто. Наукові положення, результати та висновки є власним інтелектуальним внеском автора.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ СТАНУ ВИВЧЕННЯ ПРОБЛЕМИ

1.1 Аналіз предметної області та особливостей e-commerce-платформ для домашнього текстилю

Сучасний ринок електронної комерції в Україні демонструє стійке зростання попри турбулентність зовнішнього середовища. Ринок домашнього текстилю займає одну з ключових ніш у структурі онлайн-продажів товарів для дому, що зумовлено специфікою споживчої поведінки: покупці все частіше надають перевагу зручності дистанційного вибору постільної білизни, рушників, штор, покривал та супутньої продукції замість традиційного відвідування фізичних магазинів. Підприємство «Fiona Home», яке функціонує через платформу fionahome.com.ua, є типовим представником сегменту спеціалізованих інтернет-магазинів домашнього текстилю середнього цінового діапазону, орієнтованих на вітчизняного споживача.

Предметна область e-commerce охоплює сукупність процесів, пов'язаних із організацією продажу товарів або послуг через електронні канали комунікації з використанням мережі Інтернет. У контексті домашнього текстилю ця предметна область набуває специфічних характеристик, що визначають особливі вимоги до інформаційної системи платформи. По-перше, товари цієї категорії мають яскраво виражену візуальну складову – колір, фактура, орнамент тканини є визначальними факторами при ухваленні рішення про купівлю, тому якість зображень, точність колірної передачі та детальність опису матеріалів є критично важливими параметрами для торговельного майданчика. По-друге, асортиментна матриця магазину домашнього текстилю є надзвичайно широкою та варіативною: один артикул постільної білизни може мати десятки варіацій за розміром, кольором і комплектацією, що висуває підвищені вимоги до системи управління

товарними позиціями та атрибутами. По-третє, сезонність споживчого попиту і регулярне оновлення колекцій потребують гнучкого управління каталогом та цінами [3, с. 24].

Інформаційна система e-commerce-платформи fionahome.com.ua (рис. 1.1), побудована на базі системи управління контентом OpenCart.

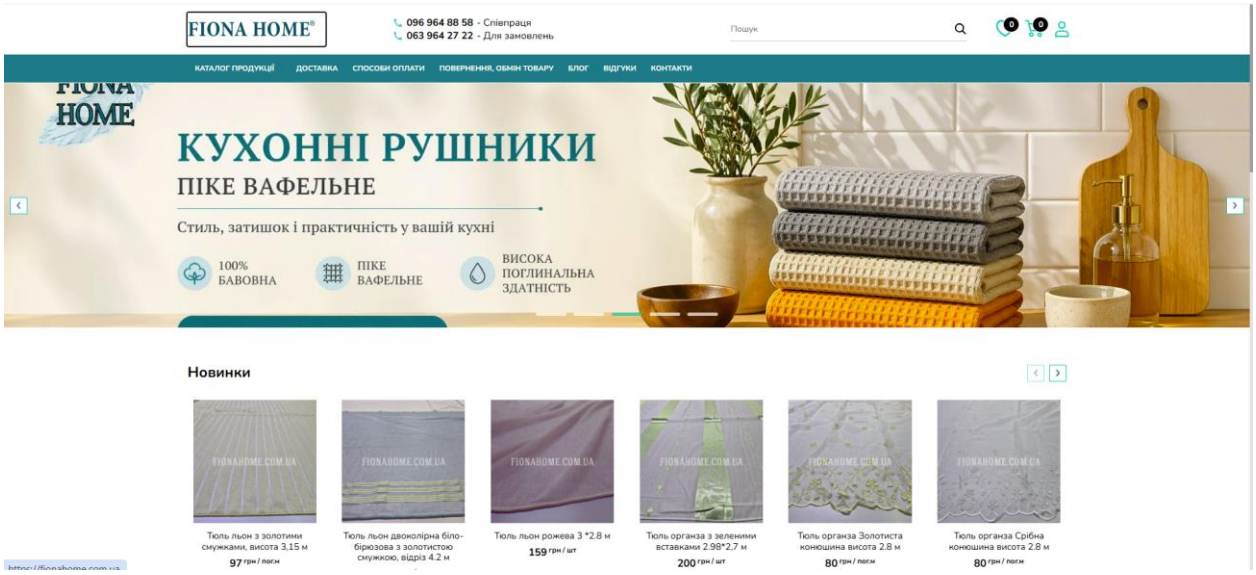


Рис. 1.1 – Головна сторінка fionahome

Будь-який інтернет-магазин як інформаційна система складається з кількох рівнів: клієнтської частини (frontend), серверної частини (backend), бази даних та зовнішніх інтеграцій. Клієнтська частина відповідає за відображення інтерфейсу користувача – каталогу товарів, сторінок продуктів, кошика, форм оформлення замовлень. Серверна частина забезпечує бізнес-логіку: обробку замовлень, управління запасами, розрахунок вартості доставки, авторизацію користувачів. База даних зберігає структуровану інформацію про товари, клієнтів, замовлення та налаштування системи. Зовнішні інтеграції охоплюють платіжні шлюзи, сервіси доставки, поштові сервіси та аналітичні системи (табл. 1.1) [1, с. 47].

Таблиця 1.1 – Характеристика рівнів архітектури e-commerce-платформи fionahome.com.ua

Рівень	Компонент	Технологія / Рішення	Функціональне призначення
Frontend	Шаблон	OpenCart Theme (Bootstrap)	Відображення каталогу, продуктових сторінок, кошика
Backend	CMS	OpenCart 3.x	Обробка бізнес-логіки, маршрутизація
База даних	СУБД	MySQL 5.7+	Зберігання товарів, замовлень, клієнтів
Інтеграції	Оплата	Оплата на реквізити	Онлайн-оплата замовлень
Інтеграції	Доставка	Нова Пошта API	Розрахунок вартості та оформлення відправлень
Адміністрування	Панель	OpenCart Admin	Управління каталогом, замовленнями, звітами

Ринок платформ для електронної комерції є достатньо конкурентним. Серед найпоширеніших рішень, що використовуються для побудови інтернет-магазинів різного масштабу, слід виділити WooCommerce (плагін для WordPress), Magento, Shopify, PrestaShop та OpenCart. Кожна з цих платформ має власну архітектуру, модель розповсюдження (SaaS або self-hosted), технічні вимоги та екосистему розширень (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика популярних CMS-платформ для e-commerce

Критерій	OpenCart	WooCommerce	Magento	PrestaShop	Shopify
Модель розповсюдження	Self-hosted, Open Source	Self-hosted, Open Source	Self-hosted / Cloud	Self-hosted, Open Source	SaaS
Вартість ліцензії	Безкоштовна	Безкоштовна	Безкоштовна / платна	Безкоштовна	Від $29/міс
Складність налаштування	Низька	Середня	Висока	Середня	Низька
Масштабованість	Середня	Середня	Висока	Середня	Висока
Спільнота в Україні	Значна	Велика	Мала	Мала	Середня
Вбудоване управління атрибутами товарів	Так	Через плагіни	Так	Так	Обмежено
Типова аудиторія	Малий та середній бізнес	Малий бізнес	Великий бізнес	Малий та середній бізнес	Будь-який

Вибір OpenCart як платформи для fionahome.com.ua обумовлений рядом прагматичних факторів. CMS OpenCart є безкоштовним відкритим рішенням з активною україномовною спільнотою, що спрощує пошук розробників і технічної підтримки. Система має зрозумілу адміністративну панель, що дозволяє менеджерам магазину самостійно управляти каталогом без глибоких технічних знань. Як свідчить аналіз інтерфейсу, панель управління (рис. 1.2) містить ключові розділи для роботи з каталогом товарів, модулями/розширеннями, дизайном, продажами, клієнтами, блогом, маркетинговими інструментами, системними налаштуваннями та звітами. Особливу увагу в контексті тестування заслуговує блок управління замовленнями, де відображаються актуальні дані про замовлення (номер, спосіб оплати, статус, дату додавання, місто, суму) та надається можливість змінювати їхній статус.

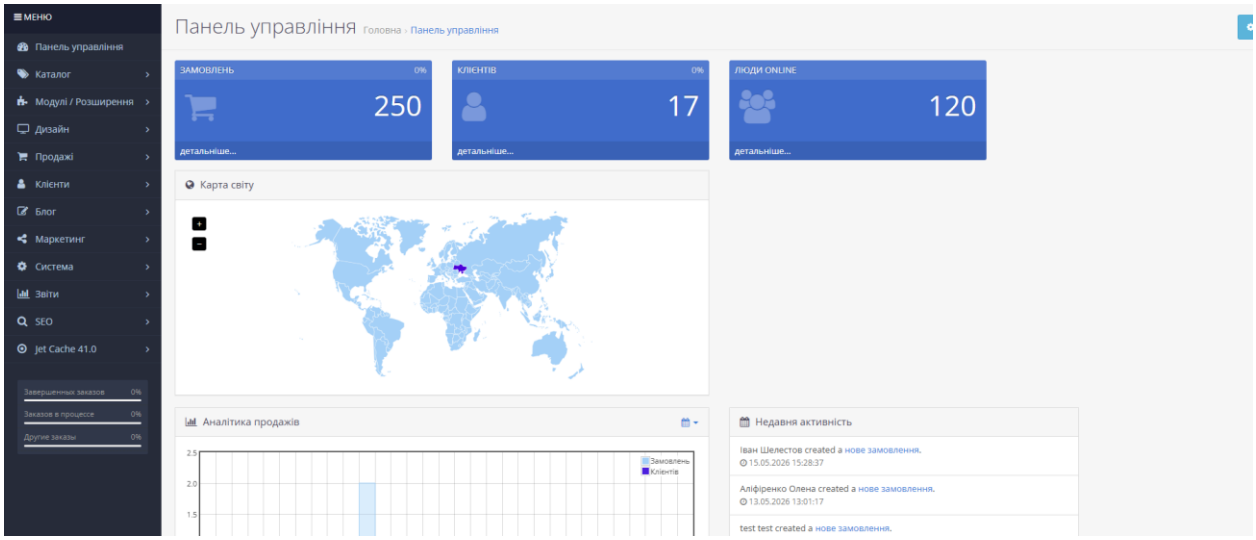


Рис. 1.2 – Адмін-панель сайту

Вбудована система атрибутів і опцій товарів добре підходить для специфіки домашнього текстилю з його варіативністю за розмірами і кольорами. Нарешті, платформа має зрілу екосистему модулів, зокрема для інтеграції з українськими платіжними системами та службами доставки [2, с. 112].

Розглядаючи специфіку предметної області домашнього текстилю як сегменту електронної комерції, слід звернути увагу на кілька ключових функціональних блоків інформаційної системи, що потребують особливої уваги під час тестування. Першим є каталог товарів із системою фільтрації, яка дозволяє покупцям знаходити продукцію за матеріалом, розміром, кольором, виробником і ціновим діапазоном. Зокрема, fionahome спеціалізується на тканинах, в асортименті сайту є: тюль, штори, рушники, декор, домашній текстиль, фурнітура та багато іншого, як можна побачити на рис. 1.3.

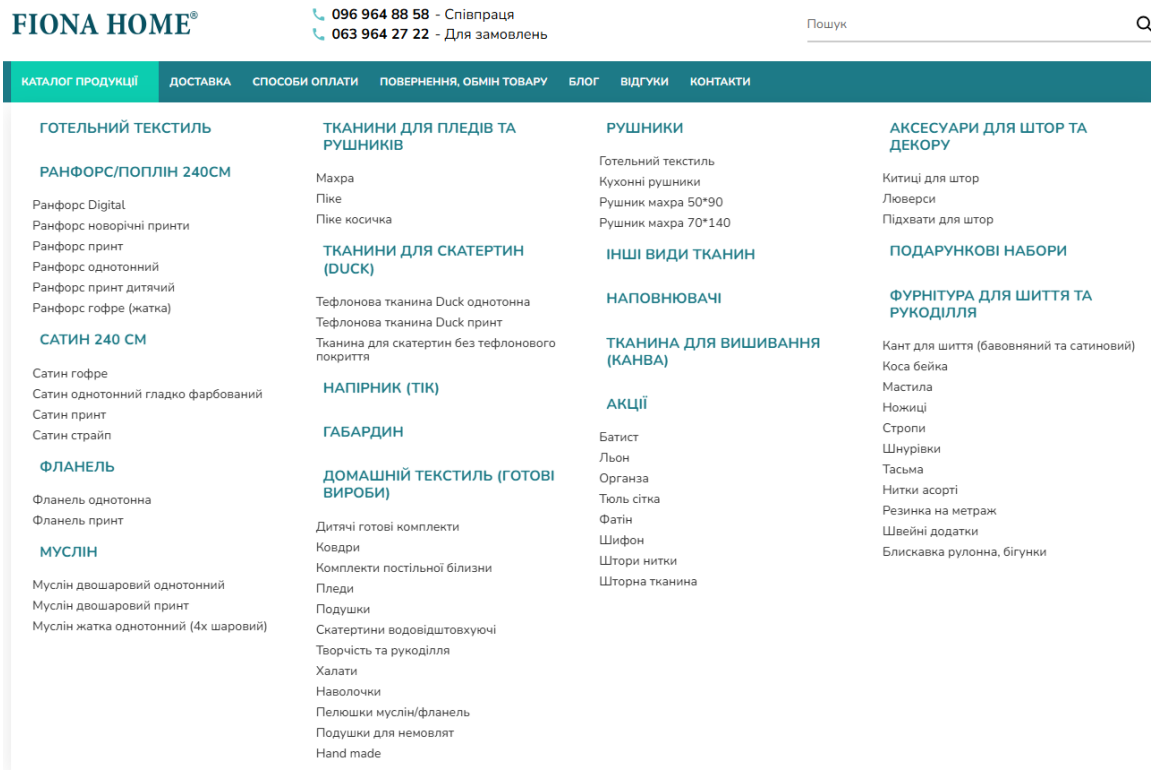


Рис. 1.3 – Каталог товарів fionahome

Другим є картка товару з детальним описом складу тканини, розмірними таблицями, галереєю зображень та блоком вибору опцій. Картка товару на fionahome (рис. 1.3) реалізована з урахуванням специфіки реалізації тканин. Крім стандартних елементів (назва, ціна, зображення, кнопка «Купити»), сторінка містить важливі для текстильної продукції характеристики: ширину полотна, щільність, склад, країну-виробника та сертифікацію. Особливістю картки є деталізація умов продажу тканини – ціна вказана за погонний метр,

мінімальний відріз становить 0,5 м, крок нарізки – 10 см, що є критично важливим для коректного розрахунку вартості замовлення. Також на сторінці реалізовано блоки з інформацією про доставку (Нова Пошта, Укрпошта), способи оплати (готівка при отриманні, безготівковий розрахунок, картки Visa/Mastercard) та умови гарантії/повернення.

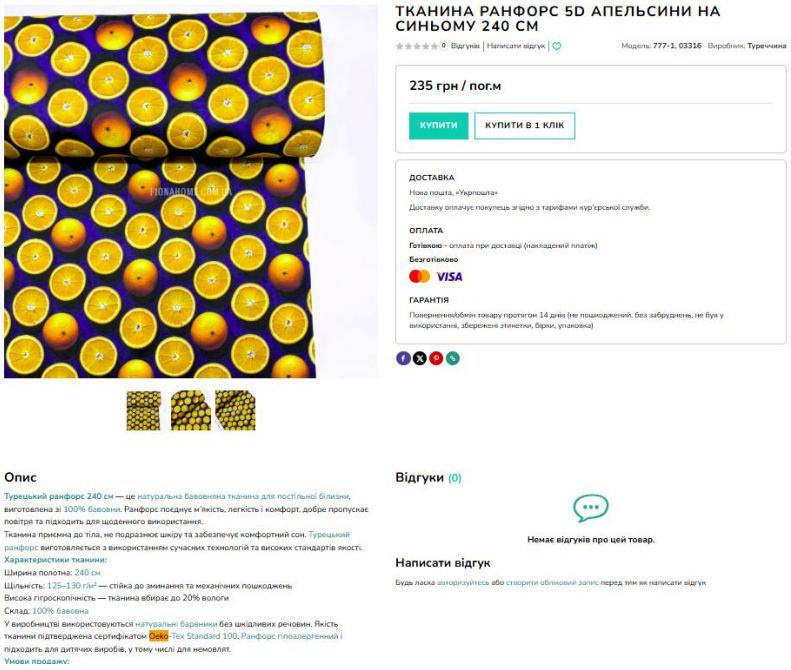


Рис. 1.4 – Приклад картки товару на сайті

Третім є кошик (рисунок 1.4) і процес оформлення замовлення (checkout), що включає вибір способу доставки, введення адреси та оплату.

Рис. 1.5 – Кошик замовлень

Четвертим є особистий кабінет покупця з історією замовлень (рис. 1.6). П'ятим є адміністративна панель з управлінням залишками, цінами та замовленнями, а шостим – особистий кабінет (рис. 1.6) [4].

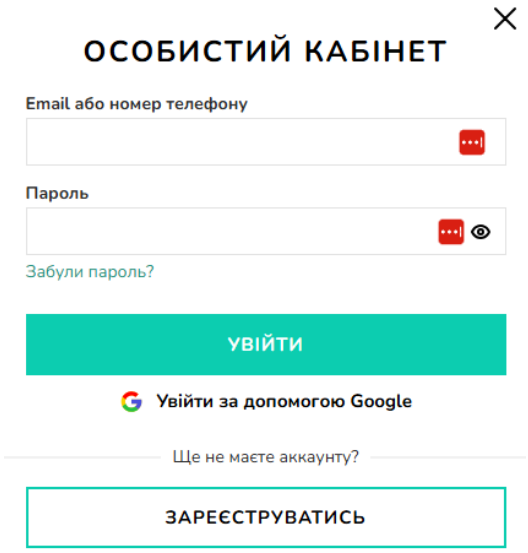


Рис. 1.6 – Форма входу в особистий кабінет

Основні елементи платформи відображено на рис. 1.7.

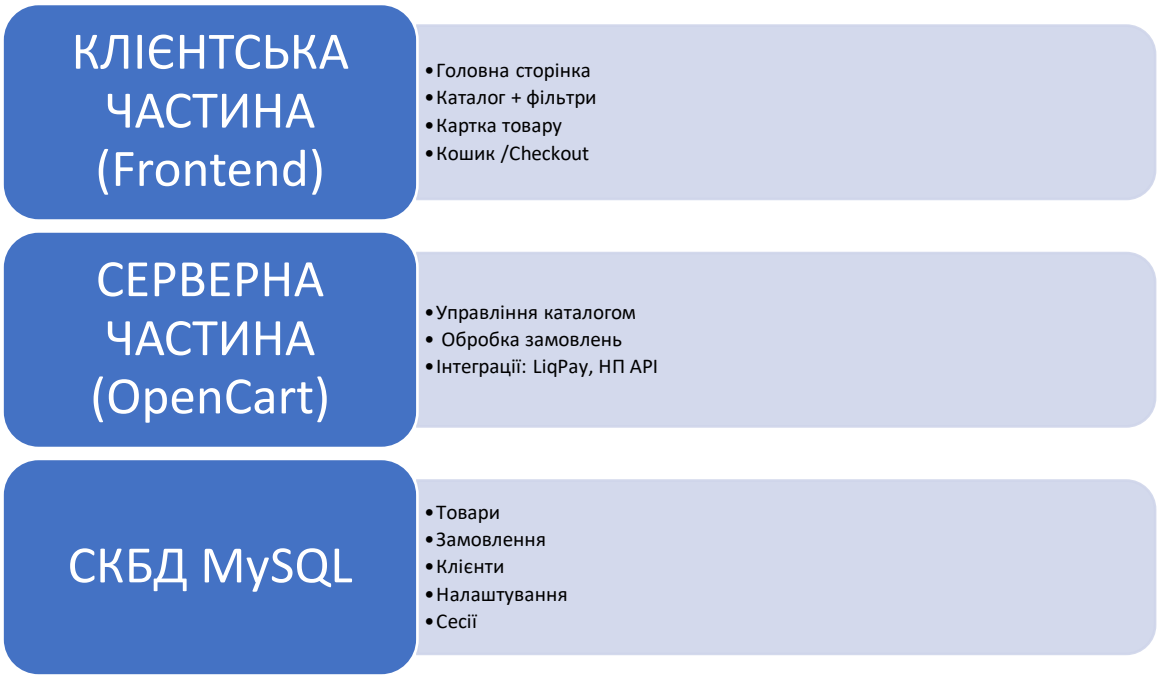


Рис. 1.7 – Функціональна модель платформи fionahome.com.ua

Важливим аспектом ІС є специфіка користувацьких сценаріїв. Типова сесія покупця на платформі домашнього текстилю значно довша порівняно з магазинами інших категорій, оскільки споживач схильний детально розглядати зображення, читати опис складу тканини, порівнювати варіанти і нерідко повертатися до вибраних позицій кілька разів перед остаточним рішенням. Це означає, що критичними для якості платформи є не лише функціональна коректність операцій, але й швидкість завантаження зображень високої роздільної здатності, стабільність стану кошика при тривалих сесіях та коректність роботи фільтрів каталогу при великій кількості SKU [5].

Проведений аналіз свідчить про те, що e-commerce-платформа для домашнього текстилю є складною багаторівневою інформаційною системою, якість функціонування якої прямо впливає на комерційну ефективність підприємства. Будь-яка помилка у роботі кошика, неправильне відображення цін чи недоступність платіжного шлюзу безпосередньо призводить до фінансових втрат через незавершені замовлення. Тому систематичне тестування усіх компонентів платформи є не технічною формальністю, а необхідною умовою сталого функціонування бізнесу.

1.2 Аналіз методів та засобів тестування вебзастосунків

Методологічну основу тестування вебзастосунків складають кілька базових концепцій. Перша – це класифікація за рівнем деталізації або так звана «піраміда тестування», введена Майком Коном. Відповідно до цієї концепції, тести поділяються на модульні (unit tests), інтеграційні (integration tests) та наскрізні або end-to-end тести (E2E tests). Модульні тести перевіряють окремі функції або методи в ізоляції від решти системи, є найшвидшими у виконанні та найбільш чисельними. Інтеграційні тести перевіряють взаємодію між кількома компонентами системи – наприклад, між PHP-контролером і СКБД MySQL в архітектурі OpenCart. Наскрізні тести імітують реальні дії користувача в браузері від початку до кінця сценарію, наприклад: відкриття сторінки каталогу, вибір товару, додавання до кошика і оформлення замовлення (рис. 1.8) [6, с. 14].

Рис. 1.8 – Піраміда тестування для вебзастосунків e-commerce

Друга базова концепція – поділ на функціональне та нефункціональне тестування. Функціональне тестування перевіряє, чи виконує система ті дії, які

вона повинна виконувати згідно зі специфікацією: чи правильно розраховується вартість замовлення, чи коректно застосовується знижка, чи надсилається підтвердження на e-mail. Нефункціональне тестування оцінює характеристики системи поза межами бізнес-логіки: продуктивність під навантаженням, безпеку, доступність для людей з обмеженими можливостями, сумісність з різними браузерами.

Третя концепція – поділ на ручне (manual) та автоматизоване (automated) тестування. Ручне тестування передбачає, що тестувальник самостійно виконує дії в системі і порівнює результат з очікуваним. Воно ефективне для дослідницького тестування, перевірки нових функцій і оцінки зручності використання (UX). Автоматизоване тестування передбачає написання скриптів або тест-кейсів, що виконуються автоматично і можуть запускатися многоразово без участі людини [8, с. 87].

Розглянемо детальніше ключові методи тестування, релевантні для платформ електронної комерції.

Функціональне тестування за методом «чорної скриньки» (black-box testing) є базовим підходом, при якому тестувальник не має доступу до вихідного коду і перевіряє систему виключно через інтерфейс. Для e-commerce-платформи це означає виконання тест-кейсів, що охоплюють реєстрацію і авторизацію, пошук товарів, роботу фільтрів, додавання товарів до кошика, застосування купонів, вибір доставки і оформлення замовлення. Протилежним є підхід «білої скриньки» (white-box testing), при якому тестувальник має доступ до коду і будує тест-кейси на основі аналізу логіки програми, покриваючи різні гілки виконання. Гібридний підхід «сірої скриньки» (grey-box testing) поєднує обидва методи і є найбільш поширеним на практиці.

Тестування продуктивності (performance testing) охоплює кілька підвидів. Навантажувальне тестування (load testing) перевіряє поведінку системи при очікуваному рівні навантаження – кількості одночасних

користувачів, що є типовою для пікових годин. Стресове тестування (stress testing) перевіряє поведінку системи в умовах, що перевищують нормальні межі, щоб визначити точку відказу. Тестування стабільності (soak testing) перевіряє систему під нормальним навантаженням протягом тривалого часу для виявлення витоків пам'яті та деградації продуктивності [9].

Тестування безпеки (security testing) є особливо важливим для платформ електронної комерції, оскільки вони обробляють персональні дані користувачів та фінансові транзакції. Основні напрями включають перевірку на SQL-ін'єкції, XSS-атаки (міжсайтовий скриптинг), CSRF-уразливості, небезпечну пряму посилання на об'єкти (IDOR) та перевірку коректності авторизації і управління сесіями. OpenCart, як зріла платформа, має вбудовані механізми захисту від типових атак, однак кастомні модулі можуть вносити нові вразливості, що потребує відповідної перевірки.

Крос-браузерне тестування (cross-browser testing) забезпечує коректне відображення і функціонування платформи в різних браузерах – Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.

Тестування зручності використання (usability testing) оцінює, наскільки легко і зрозуміло для цільової аудиторії користуватися платформою. Воно включає аналіз навігації, зрозумілість форм і підказок, читабельність текстів, розмір і розташування кнопок на мобільних пристроях [7, с. 231].

API-тестування набуває значущості для платформ з зовнішніми інтеграціями. fionahome.com.ua використовує API Нової Пошти для розрахунку вартості та оформлення відправлень, а також API платіжних систем. Коректність роботи цих інтеграцій безпосередньо впливає на повноту функціональності процесу замовлення (табл. 1.3).

Таблиця 1.3 – Класифікація методів тестування вебзастосунків та їх застосування в e-commerce

Метод тестування	**Рівень**	**Мета**	**Пріоритет для e-commerce**	**Тип**

Функціональне (black-box)	Система	Перевірка бізнес-логіки	Критичний	Ручне / Авто
Модульне (unit)	Компонент	Ізольована перевірка функцій	Середній	Авто
Інтеграційне	Підсистема	Взаємодія компонентів	Високий	Авто
Регресійне	Система	Стабільність при змінах	Критичний	Авто
Навантажувальне	Система	Поведінка при навантаженні	Високий	Авто
Крос-браузерне	UI	Сумісність браузерів	Високий	Ручне / Авто
Мобільне	UI	Адаптивність	Критичний	Ручне / Авто
Безпека	Система	Захист даних	Критичний	Авто
API	Інтеграції	Коректність інтеграцій	Високий	Авто
Юзабіліті	UX	Зручність використання	Середній	Ручне
Приймальне (UAT)	Система	Відповідність вимогам	Критичний	Ручне

Окрему увагу слід приділити концепції тестування на основі ризиків (risk-based testing). Цей підхід передбачає пріоритизацію тестових зусиль відповідно до ймовірності та впливу потенційних дефектів. Для e-commerce-платформи найвищий пріоритет мають ті частини системи, відмова яких призводить до прямих фінансових втрат: процес checkout, платіжні інтеграції, відображення цін і наявності товарів. Менш критичними є, наприклад, сторінки статичного контенту або розділи блогу [10, с. 56].

1.3 Аналіз та вибір інструментів тестування на базі CMS OpenCart

Вибір інструментів тестування для конкретної платформи є відповідальним інженерним рішенням, що залежить від технологічного стеку системи, доступних ресурсів, рівня технічної зрілості команди і специфіки вимог до якості. OpenCart є PHP-застосунком зі стандартною MVC-архітектурою і СКБД MySQL, що визначає технологічний контекст для вибору

тестових інструментів. Розглянемо категорії інструментів, релевантних для тестування вебзастосунку fionahome. Інструменти для функціонального та наскрізного тестування:

Selenium WebDriver є індустріальним стандартом для автоматизованого тестування вебінтерфейсів. Інструмент підтримує мови програмування Java, Python, PHP, C#, Ruby та JavaScript, дозволяє керувати браузером програмно, імітуючи дії реального користувача. Для OpenCart на Selenium можна реалізувати автоматизовані тест-кейси для всіх ключових сценаріїв: пошук товарів, додавання до кошика, заповнення форм оформлення замовлення. Недоліком Selenium є відносно повільне виконання тестів та необхідність налаштування WebDriver для кожного браузера [11].

Playwright є сучаснішою альтернативою Selenium, розробленою Microsoft.

Cypress є ще одним популярним інструментом для E2E-тестування, орієнтованим виключно на JavaScript і роботу в браузерах на базі Chromium.

Інструменти для тестування продуктивності:

Apache JMeter є безкоштовним відкритим інструментом для навантажувального тестування.

Google Lighthouse є вбудованим в Chrome DevTools інструментом для аудиту вебсторінок за параметрами продуктивності (Core Web Vitals), доступності, SEO та дотримання найкращих практик. Він генерує докладні звіти з конкретними рекомендаціями щодо оптимізації. Для fionahome.com.ua, де сторінки товарів містять численні великі зображення тканин, аудит продуктивності за допомогою Lighthouse є особливо актуальним [12].

Інструменти для тестування безпеки:

OWASP ZAP (Zed Attack Proxy) є безкоштовним інструментом для пошуку вразливостей у вебзастосунках, розробленим організацією OWASP. Він може виступати в ролі проксі між браузером і сервером, аналізуючи трафік і автоматично шукаючи типові вразливості: SQL-ін'єкції, XSS, CSRF. Burp

Suite Community Edition є альтернативним інструментом для ручного та напівавтоматичного тестування безпеки.

Інструменти для модульного та інтеграційного тестування PHP:

PHPUnit є стандартним фреймворком для модульного тестування PHP-коду. Він підтримує написання тест-класів, що перевіряють окремі методи і класи. В архітектурі OpenCart PHPUnit можна використовувати для тестування кастомних модулів і контролерів.

Інструменти для API-тестування:

Postman є найпоширенішим інструментом для ручного та автоматизованого тестування HTTP API. Він дозволяє відправляти запити до ендпоінтів API, перевіряти відповіді за допомогою JavaScript-скриптів (assertions) та організовувати тести у колекції з можливістю запуску через CLI за допомогою Newman. Для тестування інтеграції OpenCart з API Нової Пошти і платіжних систем Postman є ефективним рішенням.

Інструменти для управління тестуванням:

TestRail є хмарною системою управління тестами, що дозволяє організовувати тест-кейси в плани, відстежувати виконання та генерувати звіти. Як безкоштовна альтернатива може використовуватися Qase або навіть структуровані таблиці в Google Sheets. Для документування та відстеження дефектів широко застосовуються Jira та GitHub Issues (табл. 1.4) [13].

Таблиця 1.4 – Порівняльний аналіз інструментів тестування для OpenCart-платформи

Інструмент	**Категорія**	**Мова**	**Ліцензія**	**Складність**	**Придатність для OpenCart**
Selenium	E2E / UI	Java, Python, PHP	Open Source	Середня	Висока
Playwright	E2E / UI	JS, Python, Java	Open Source	Середня	Висока
Cypress	E2E / UI	JavaScript	Open Source	Низька	Висока
Pytest	Unit / E2E	Python	Open Source	Низька	Висока

Apache JMeter	Performance	Java (GUI)	Open Source	Висока	Висока
k6	Performance	JavaScript	Open Source	Низька	Висока
Google Lighthouse	Performance / SEO	– (CLI/браузер)	Open Source	Низька	Висока
OWASP ZAP	Security	– (GUI)	Open Source	Середня	Висока
Postman	API	JavaScript	Freemium	Низька	Висока
TestRail	Test Management	–	Платна	Середня	Висока

Здійснюючи вибір інструментів для конкретної задачі дослідження, слід враховувати не лише технічні характеристики кожного засобу, але і їх сукупну здатність покрити повний спектр тестових потреб платформи fionahome.com.ua. Попередній аналіз дозволив виявити наступні пріоритетні напрями тестування: функціональна коректність сценарію покупки (каталог → картка товару → кошик → checkout), коректність роботи фільтрів і пошуку, продуктивність при завантаженні сторінок з великою кількістю зображень, крос-браузерна та мобільна сумісність, безпека форм і механізму авторизації, коректність роботи інтеграцій з API Нової Пошти та платіжними системами [14].

Для покриття функціональних і наскрізних тест-кейсів найбільш доцільним є використання Selenium WebDriver у зв'язці з мовою Python, оскільки цей стек є достатньо зрілим і документованим, має широку підтримку в україномовній спільноті і дозволяє реалізувати тест-кейси будь-якої складності для OpenCart-інтерфейсу.

Для тестування продуктивності використовується Google Lighthouse як засіб швидкого аудиту і JMeter для симуляції навантаження при пікових відвідуваннях. Тестування безпеки проводиться за допомогою OWASP ZAP. API-тестування виконується через Postman з автоматизованими колекціями. Модульне тестування окремих компонентів системи та автоматизовані перевірки реалізовано за допомогою pytest (рис. 1.9).

Рис. 1.9 – Схема організації тестування платформи fionahome.com.ua

Особливістю архітектури OpenCart є те, що вона реалізує патерн MVC і має чітко виражені рівні – моделі (взаємодія з базою даних), контролери (бізнес-логіка та маршрутизація) і представлення (шаблони). Це дозволяє організувати тестування на кожному з цих рівнів незалежно. Моделі можна тестувати за допомогою PHPUnit з мок-об'єктами бази даних. Контролери можна тестувати через HTTP-запити за допомогою Postman або спеціалізованих PHP-бібліотек. Представлення тестуються через UI-автоматизацію за допомогою Selenium або Playwright (табл. 1.5) [15].

Таблиця 1.5 – Матриця відповідності методів тестування до рівнів архітектури OpenCart

Рівень OpenCart	**Технологія**	**Метод тестування**	**Інструмент**
Модель (Model)	PHP + MySQL	Модульне	Pytest
Контролер (Controller)	PHP	Інтеграційне / API	Postman, Pytest
Представлення (View)	Twig/PHP шаблони	UI / Функціональне	Selenium, Playwright
База даних	MySQL	Інтеграційне	SQL-запити, Pytest
Зовнішні API	REST/SOAP	API	Postman

Вся система	–	E2E / UAT	Selenium WebDriver, Pytest

Розглянувши весь спектр наявних інструментів і методів тестування, а також провівши їх відображення на специфіку архітектури OpenCart і предметної області домашнього текстилю, можна сформулювати такий висновок. Для комплексного забезпечення якості платформи fionahome.com.ua доцільно використовувати багаторівневий підхід, що поєднує автоматизоване функціональне тестування на основі Selenium WebDriver та Python, навантажувальне тестування за допомогою Google Lighthouse і JMeter, тестування безпеки за допомогою OWASP ZAP, API-тестування засобами Postman та модульне тестування PHP-компонентів за допомогою pytest. Такий підхід забезпечує покриття всіх рівнів системи і всіх ключових аспектів якості, що відповідає вимогам до дослідження, сформульованим у технічному завданні на кваліфікаційну роботу.

РОЗДІЛ 2. ОПИС МОДЕЛІ ТА МЕТОДІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ E-COMMERCE-ПЛАТФОРМИ НА БАЗІ CMS OPENCART

2.1 Розробка концептуальної моделі системи

Концептуальна модель інформаційної системи є абстрактним описом структури і поведінки системи, що відображає суттєві характеристики предметної області незалежно від конкретної технологічної реалізації. Для платформи fionahome.com.ua концептуальна модель повинна відображати не лише статичну структуру даних, але і динаміку бізнес-процесів – від перегляду каталогу до завершення замовлення та його відправки.

Розробка концептуальної моделі розпочинається з ідентифікації ключових суб'єктів (акторів) та об'єктів системи. Акторами платформи є: незареєстрований відвідувач (гість), зареєстрований покупець, адміністратор магазину та менеджер замовлень. Кожен з акторів взаємодіє з системою за різними сценаріями і має різний рівень доступу до функцій платформи (рис. 2.1) [18, с. 143].

Рис. 2.1 – Діаграма варіантів використання (Use Case) для платформи fionahome.com.ua

Центральним бізнес-процесом платформи є процес здійснення покупки. Концептуально він розбивається на декілька послідовних фаз: фаза дослідження (browsing) – користувач переглядає каталог, застосовує фільтри, ознайомлюється з описом товару; фаза вибору – користувач обирає конкретну опцію товару (розмір, колір) і додає її до кошика; фаза оформлення (checkout) – користувач вводить контактні дані, обирає спосіб і адресу доставки, застосовує купон знижки за наявності; фаза оплати – здійснення транзакції через платіжний шлюз або вибір оплати при отриманні; фаза підтвердження – система відправляє підтвердження замовлення на електронну пошту і формує запис у базі даних.

Для платформи домашнього текстилю особливо важливою є коректна реалізація фази вибору, оскільки один товар – наприклад, комплект постільної білизни – може мати від чотирьох до двадцяти варіацій за розміром (євро, полуторний, двоспальний, сімейний) і до десяти варіацій за кольором або орнаментом. Система опцій OpenCart реалізує цю варіативність через механізм атрибутів і опцій товару, де кожна комбінація може мати власну ціну, залишок на складі та артикул (SKU) (рис. 2.2) [16, с. 76].

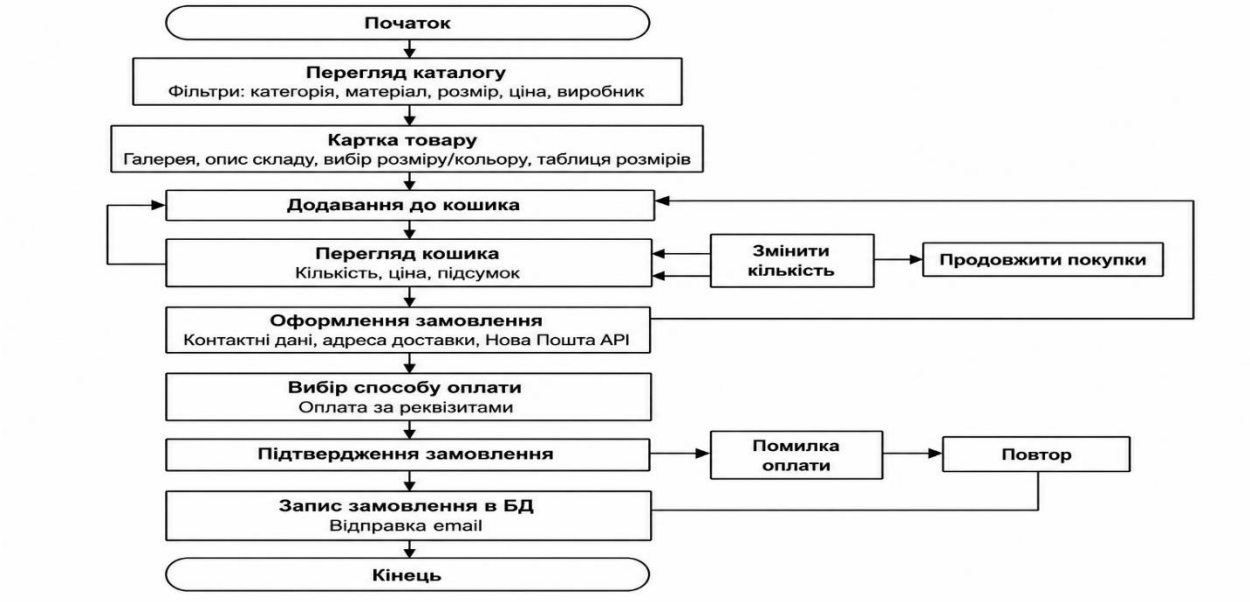


Рис. 2.2 – Концептуальна блок-схема бізнес-процесу здійснення покупки на fionahome.com.ua

Концептуальна модель системи охоплює не лише клієнтський сценарій, але і внутрішні адміністративні процеси. Адміністративна панель OpenCart на fionahome.com.ua забезпечує управління каталогом товарів – додавання нових позицій, редагування описів, оновлення цін і залишків, публікацію акцій. Менеджер замовлень обробляє вхідні замовлення, змінює їх статуси, формує відправлення через Нову Пошту та комунікує з покупцями через вбудований механізм сповіщень.

Концептуальна модель також фіксує межі системи і точки її взаємодії із зовнішніми сервісами. Для fionahome.com.ua такими зовнішніми сервісами є API Нової Пошти (розрахунок вартості доставки, перелік відділень і поштоматів, оформлення ТТН) та сервіси Google Analytics для збору статистики відвідуваності й аналізу поведінки користувачів на сайті(табл. 2.1) [17, с. 138].

Таблиця 2.1 – Специфікація зовнішніх інтеграцій платформи fionahome.com.ua

Зовнішній сервіс	**Тип інтеграції**	**Протокол**	**Призначення**	**Критичність**
Нова Пошта API	REST API	HTTPS/JSON	Перелік відділень, розрахунок вартості, оформлення ТТН	Критична
Google Analytics 4	JavaScript SDK	HTTPS	Збір статистики відвідуваності та аналіз поведінки користувачів	Некритична
SMTP-сервер	SMTP	TLS	Надсилання підтверджень замовлень	Висока

Архітектурно система fionahome.com.ua реалізована за класичною трирівневою моделлю клієнт-сервер. Рівень представлення (presentation tier) реалізований через HTML/CSS/JavaScript-шаблони OpenCart, що рендеряться на сервері і доставляються браузеру клієнта. Рівень бізнес-логіки (logic tier)

реалізований у PHP-коді OpenCart і включає контролери, моделі та бібліотеки. Рівень даних (data tier) реалізований на СУБД MySQL і містить всю структуровану інформацію системи (рис. 2.3) [19].

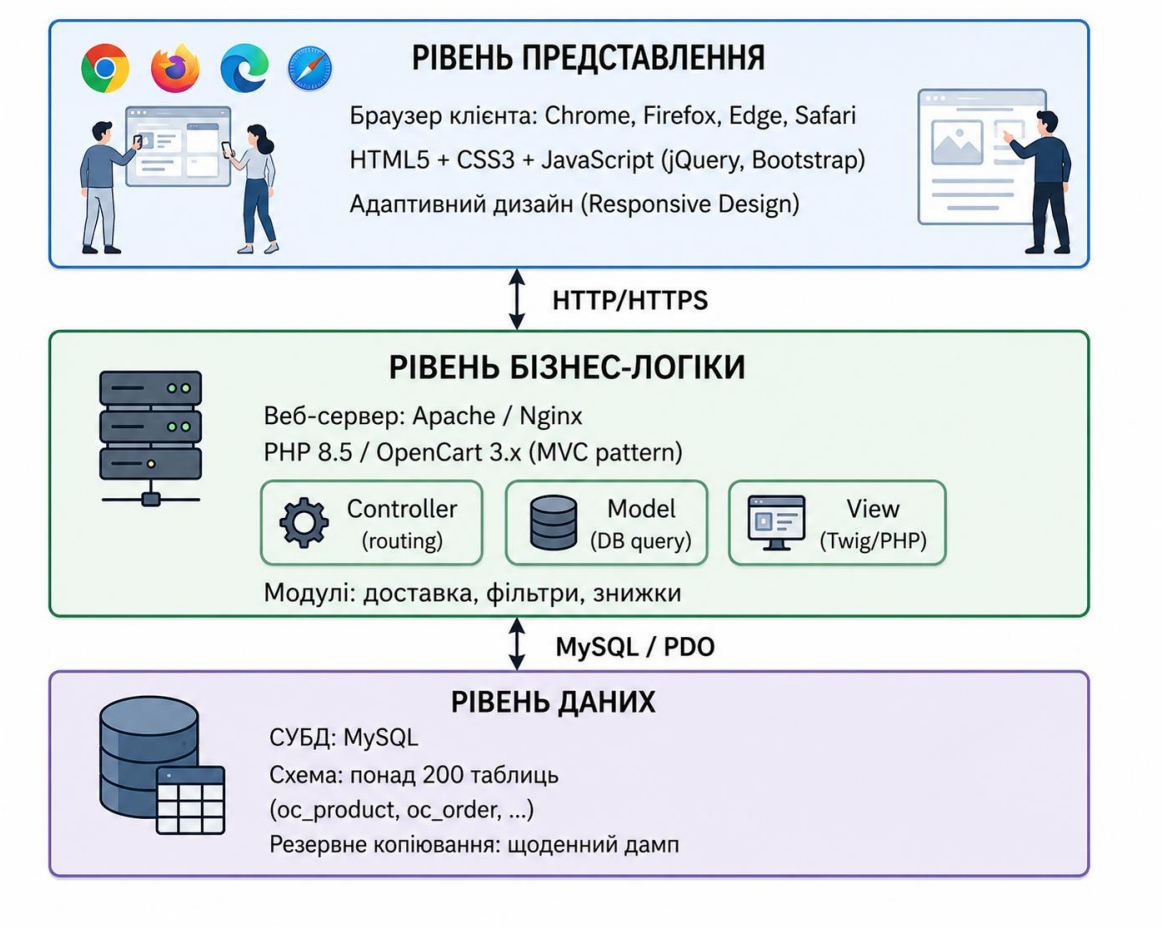


Рис. 2.3 – Трирівнева архітектура платформи fionahome.com.ua

Важливим аспектом концептуальної моделі є опис станів замовлення, оскільки саме коректна зміна статусів є предметом ключових тест-кейсів. OpenCart підтримує розгалужений граф станів замовлення, що налаштовується адміністратором. На платформі fionahome.com.ua використовуються такі статуси: «Очікує підтвердження», «Підтверджено», «В обробці», «Передано на відправку», «Відправлено», «Доставлено», «Скасовано», «Повернення». Кожен перехід між статусами може супроводжуватися автоматичним відправленням email-сповіщення покупцю з відповідним шаблоном повідомлення (рис. 2.4).

Рис. 2.4 – Діаграма станів замовлення на платформі fionahome.com.ua

Концептуальна модель системи також включає опис ролей і привілеїв доступу. OpenCart реалізує рольову модель управління доступом (RBAC) в адміністративній панелі, де кожній ролі відповідає набір дозволених дій [20]. Для fionahome.com.ua доцільно розмежувати роль адміністратора (повний доступ) і ролі менеджера (обробка замовлень без доступу до налаштувань системи), що є важливим аспектом не лише з точки зору безпеки, але і в контексті тестування правильності реалізації прав доступу.

2.2 ER-модель та нормалізація відношень бази даних

База даних є фундаментальною складовою будь-якої інформаційної системи. У OpenCart база даних містить близько сімдесяти таблиць з префіксом `oc_`, що охоплюють всі аспекти функціонування платформи.

Проектування і розуміння структури цієї бази даних є необхідною умовою для грамотного складання інтеграційних тест-кейсів і верифікації коректності збереження даних після виконання операцій через інтерфейс.

ER-модель (Entity-Relationship model) описує сутності предметної області, їх атрибути та зв'язки між ними. Для платформи FionaHome ключовими сутностями є товар (`fdb_product`), категорія (`fdb_category`), проміжна таблиця зв’язку товарів і категорій (`fdb_product_to_category`), покупець (`fdb_customer`), замовлення (`fdb_order`) та склад замовлення (`fdb_order_product`). Крім того, до ER-моделі включено логічну підсистему тестування, представлену сутностями `test_case`, `test_run` та `test_result`.

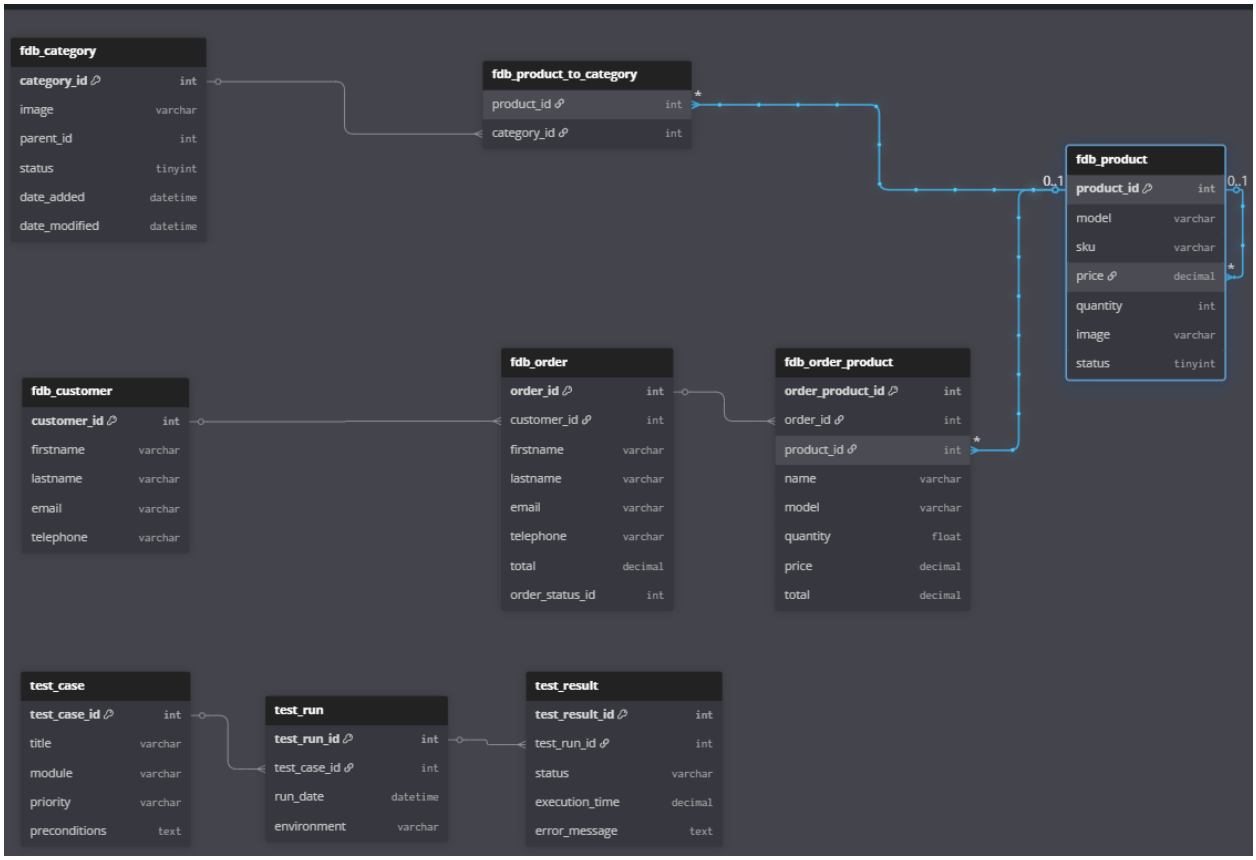


Рис. 2.5 – Спрощена ER-діаграма платформи fionahome.com.ua

ER-діаграма складається з двох частин. Перша відображає структуру бази даних інтернет-магазину, друга — структуру підсистеми автоматизованого тестування, розробленої в межах кваліфікаційної роботи.

Детальніше розглянемо структуру ключових таблиць, що безпосередньо залучені до процесу тестування критичних сценаріїв платформи fionahome.com.ua.

Центральною таблицею ER-моделі є `fdb_product`, яка містить базову інформацію про товарні позиції інтернет-магазину: ідентифікатор товару, модель, артикул, ціну, кількість на складі, основне зображення та статус активності. Таблиця `fdb_category` описує категорії товарів, а зв'язок між товарами і категоріями реалізовано через проміжну таблицю `fdb_product_to_category`. Такий підхід дозволяє одному товару належати до кількох категорій, а одній категорії містити багато товарів.

Для відображення процесу оформлення замовлень у моделі використано таблиці `fdb_customer`, `fdb_order` та `fdb_order_product`. Таблиця `fdb_customer` містить дані покупців, `fdb_order` зберігає інформацію про замовлення, а `fdb_order_product` фіксує склад кожного замовлення. Через цю таблицю реалізується зв'язок між замовленнями та товарами

Логічна підсистема тестування представлена сутностями `test_case`, `test_run` та `test_result`. `test_case` описує тестовий сценарій, `test_run` — окремий запуск тесту, а `test_result` — результат його виконання зі статусом, часом виконання та інформацією про помилку або скріншот.

Таблиця 2.2 – Структура ключових таблиць платформи fionahome.com.ua та підсистеми тестування

Таблиця	**Ключові поля**	**Призначення**	**Кількість записів (орієнтовно)**
fdb_product	product_id, model,sku, price, quantity, status	Базові дані товарів	Залежить від наповнення каталогу
fdb_category	category_id, parent_id, status	Ієрархія категорій	Залежить від структури каталогу

fdb_product_to_category	product_id, category_id	Зв'язок товарів і категорій	Залежить від кількості товарів і категорій
fdb_customer	customer_id, firstname, lastname, email	Облікові записи покупців	Зростає постійно
fdb_order	order_id, customer_id	Замовлення	Зростає постійно
fdb_order_product	order_product_id, order_id, product_id	Склад замовлень	Зростає постійно
test_case	test_case_id, title, module	Тестові сценарії	Залежить від набору тестів
test_run	test_run_id, test_case_id, run_date	Запуски тестів	Зростає під час тестування
test_result	test_result_id, test_run_id, status	Результати тестування	Зростає під час тестування

Розглянуті таблиці відповідають вимогам третьої нормальної форми (3НФ), що є стандартом для реляційних баз даних. Перша нормальна форма (1НФ) дотримується завдяки атомарності значень у полях – жодне поле не містить списків або вкладених структур. Друга нормальна форма (2НФ) дотримується через відсутність часткової функціональної залежності від складеного ключа. Третя нормальна форма (3НФ) дотримується через відсутність транзитивних залежностей: всі атрибути таблиць залежать безпосередньо від первинного ключа, а не від інших неключових атрибутів (табл. 2.3) [24, с. 84].

Таблиця 2.3 – Аналіз нормалізації ключових таблиць бази даних

Таблиця	**1НФ**	**2НФ**	**3НФ**	**Коментар**
fdb_product	✓	✓	✓	Атрибути залежать від первинного ключа product_id
fdb_category	✓	✓	✓	Ієрархія категорій реалізована через parent_id
fdb_customer	✓	✓	✓	Дані покупця залежать від customer_id
fdb_order	✓	✓	✓	Дані замовлення залежать від order_id

fdb_order_product	✓	✓	✓	Реалізує зв'язок між замовленнями та товарами
test_case	✓	✓	✓	Опис тестового сценарію залежить від test_case_id
test_run	✓	✓	✓	Дані запуску тесту залежать від test_run_id
test_result	✓	✓	✓	Дані результату тестування залежать від test_result_id

Слід зазначити, що в таблицях fdb_order і fdb_order_product присутня навмисна денормалізація, що є типовим проектним рішенням систем електронної комерції. Замість посилання на поточну назву товару або його актуальну ціну в таблиці замовлень зберігаються копії цих значень на момент оформлення покупки. Це забезпечує незмінність історії замовлень навіть у разі подальшої зміни даних товару або його вартості.

2.3 Проєктування стратегії тестування вебзастосунку

Стратегія тестування є документом вищого рівня, що визначає загальний підхід до забезпечення якості платформи: які методи тестування застосовуються, яким інструментарієм, в якій послідовності, які критерії успіху і скасування тестування. На відміну від плану тестування, що деталізує конкретні дії і строки, стратегія фіксує принципові рішення і підходи, що залишаються незмінними протягом всього циклу розробки і супроводу.

Стратегія тестування платформи fionahome.com.ua розроблена з урахуванням таких вихідних умов: платформа є вже функціонуючим комерційним ресурсом (а не розробляється з нуля), тому тестування проводиться у форматі аудиту і регресійного тестування існуючої функціональності, а не приймального тестування нової системи. Команда розробки є малою (1–2 розробники), тому стратегія має бути прагматичною і не вимагати значних ресурсів на підтримку тестової інфраструктури. Критичні для бізнесу компоненти повинні бути покриті автоматизованими тестами, тоді

як менш критичні можуть перевірятися вручну за чек-листами (рис. 2.6) [26, с. 93].

Рис. 2.6 – Схема стратегії тестування платформи fionahome.com.ua

Центральним елементом стратегії тестування є пріоритизація тест-кейсів на основі ризик-аналізу. Для платформи e-commerce ризик визначається як добуток ймовірності появи дефекту і впливу цього дефекту на бізнес. Функціональність, що обробляє платіжні транзакції, є найвищим пріоритетом, оскільки її відмова призводить до прямих фінансових втрат і підриває довіру покупців. Другим пріоритетом є процес формування і підтвердження замовлення. Третім – каталог і пошук товарів. Четвертим – особистий кабінет і управління профілем (табл. 2.4).

Таблиця 2.4 – Матриця ризиків для пріоритизації тестування fionahome.com.ua

Функціональний блок	**Ймовірність дефекту**	**Вплив на бізнес**	**Рівень ризику**	**Пріоритет тестування**
Платіжні модулі	Середня	Критичний	Критичний	P1
Процес оформлення замовлення	Середня	Критичний	Критичний	P1
Кошик (додавання, видалення, кількість)	Висока	Критичний	Критичний	P1

Продовження таблиці 2.5

Розрахунок цін і знижок	Середня	Високий	Критичний	P1
Авторизація і реєстрація	Середня	Високий	Високий	P2
Каталог і фільтри	Висока	Високий	Високий	P1
Пошук товарів	Середня	Середній	Середній	P2

Картка товару	Низька	Середній	Середній	P3
Особистий кабінет	Низька	Низький	Низький	P3
Сторінки статичного контенту	Низька	Низький	Низький	P4
Адмін-панель (управління каталогом)	Низька	Середній	Середній	P3

Проектування конкретних тест-кейсів для критичних сценаріїв базується на техніці «позитивного» і «негативного» тестування. Позитивне тестування перевіряє коректну роботу системи при введенні валідних вхідних даних і виконанні очікуваних дій. Негативне тестування перевіряє стійкість системи до некоректних даних, нестандартних дій і граничних умов (табл. 2.5) [27, с. 147].

Таблиця 2.5 – Специфікація тест-кейсів для критичного сценарію «Оформлення замовлення»

ID	**Назва тест-кейсу**	**Тип**	**Кроки**	**Очікуваний результат**	**Пріоритет**
TC-01	Перевірка відкриття сторінки оформлення замовлення	Позитивний	1. Додати товар до кошика; 2. Перейти до checkout	Сторінка оформлення замовлення успішно завантажується	P1
TC-02	Перевірка відображення полів покупця	Позитивний	1. Перейти до checkout; 2. Обрати оформлення замовлення без реєстрації	Поля для введення даних покупця відображаються коректно	P1
TC-03	Перевірка валідації обов'язкових полів	Негативний	1. Перейти до checkout; 2. Не заповнювати обов'язкові поля; 3. Спробувати продовжити оформлення	Відображаються повідомлення про помилки валідації	P1

ID	Назва тест-кейсу	Тип	Кроки	Очікуваний результат	Пріоритет
TC-04	Перевірка валідації некоректного номера телефону	Негативний	1. Перейти до checkout; 2. Ввести некоректний номер телефону; 3. Спробувати продовжити оформлення	Відображається повідомлення про помилку введення телефону	P1
TC-05	Перевірка валідації порожнього прізвища	Негативний	1. Заповнити форму покупця; 2. Не заповнювати поле «Прізвище»; 3. Спробувати продовжити оформлення	Відображається повідомлення про необхідність заповнення прізвищая	P1
TC-06	Перевірка відображення способів доставки Нової Пошти	Позитивний	1. Перейти до checkout; 2. Заповнити контактні дані; 3. Перевірити блок доставки	Відображаються доступні способи доставки Нової Пошти	P1
TC-07	Перевірка відображення полів міста та відділення Нової Пошти	Позитивний	1. Перейти до checkout; 2. Обрати доставку Новою Поштою	Відображаються поля вибору міста та відділення	P1
TC-08	Перевірка застосування купона знижки	Позитивний	1. Додати товар до кошика; 2. Ввести дійсний купон; 3. Застосувати купон	Купон успішно застосовується, сума замовлення зменшується відповідно до знижки	P1

ID	Назва тест-кейсу	Тип	Кроки	Очікуваний результат	Пріоритет

Важливою складовою процесу тестування є визначення умов, у яких виконуватимуться перевірки вебзастосунку. У межах даної роботи автоматизоване тестування проводилося безпосередньо на вебсайті fionahome.com.ua із використанням фреймворку Pytest та інструмента Selenium WebDriver. Розроблений набір тестів охоплював основні функціональні модулі системи, зокрема каталог товарів, сторінку товару, кошик, оформлення замовлення та авторизацію користувачів Оскільки окреме тестове середовище для вебзастосунку відсутнє, перевірки виконувалися на робочій версії сайту із застосуванням тестових сценаріїв, що не впливають на коректність його функціонування та не змінюють критичні дані системи. Запуск тестів здійснювався за допомогою розробленого програмного засобу, який забезпечує автоматичне виконання перевірок, збір результатів та формування підсумкового HTML-звіту. За результатами тестування було виконано 78 тестових сценаріїв, з яких 73 завершилися успішно, а 5 були автоматично позначені як skipped через відсутність або неактивність окремих елементів функціональності. Отримані результати підтвердили працездатність основних модулів вебзастосунку та ефективність розробленого набору автоматизованих тестів (рис. 2.7).

Рис. 2.7 – Схема тестових середовищ для платформи fionahome.com.ua

Автоматизована частина стратегії тестування реалізується на основі фреймворку Selenium WebDriver з мовою Python і бібліотекою pytest. Вибір pytest обумовлений його зручністю для організації тестових сценаріїв використання механізму для підготовки тестового середовища та можливість формування детальних звітів про результати виконання тестів .

Структура тестового проекту організована у відповідності до патерну Page Object Model (POM), що є загальновизнаним підходом для структурування Selenium-тестів. Суть патерну полягає у тому, що кожна сторінка вебзастосунку представлена окремим класом Python, що інкапсулює логіку взаємодії з елементами цієї сторінки. Тест-кейси при цьому оперують методами класів-сторінок, а не безпосередньо селекторами елементів. Це забезпечує виразність тестів, їх стабільність при змінах в HTML і простоту підтримки (рис. 2.8).

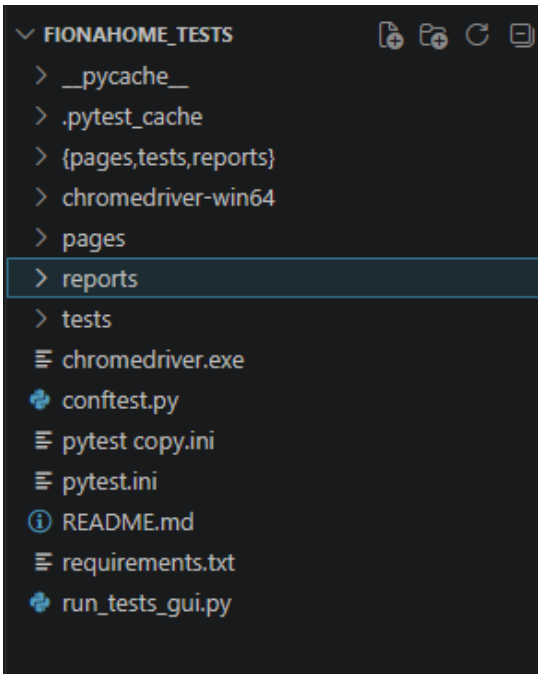


Рис. 2.8 – Структура тестового проекту за патерном Page Object Model

Приклад реалізації базового класу сторінки і класу сторінки кошика у відповідності до обраного патерну [29] наведено в додатку А.

Відповідність тест-кейсів до технік тест-дизайну відображено у таблиці 2.6.

Таблиця 2.6 – Відповідність тест-кейсів до технік тест-дизайну

Техніка тест-дизайну	**Застосування для fionahome.com.ua**	**Приклад тест-кейсу**
Еквівалентне розбиття	Поле email реєстрації	Валідний email / некоректний формат / порожнє поле
Граничний аналіз	Поле кількості товару в кошику	0, 1, 999, 1000 (максимум)
Таблиці рішень	Розрахунок доставки НП	Комбінації: передоплата / НП / поштомат / кур'єр
Переходи між станами	Стан кошика	Додавання товару → зміна кількості → видалення товару
Дослідницьке тестування	Нові модулі та інтеграції	Вільне дослідження поведінки без скриптів
Тестування на основі ризику	Пріоритизація тест-кейсів	Checkout і оплата – першочергово

Критерії успіху тестування (exit criteria) визначають умови, за яких тестування вважається завершеним і система може бути передана на наступний етап. Для платформи fionahome.com.ua встановлено такі критерії успіху тестування: відсутність тестів зі статусом Failed; успішне виконання основних функціональних перевірок; відсутність відкритих дефектів критичного та високого рівня серйозності; підтвердження працездатності ключових модулів вебзастосунку за результатами автоматизованого тестування (табл. 2.7) [30].

Таблиця 2.7 – Класифікація дефектів за рівнем серйозності (Severity) для fionahome.com.ua

Рівень	Назва	Опис	Приклад	Час реакції
S1	Blocker	Система непрацездатна, критичний функціонал недоступний	Сторінка checkout не відкривається	Негайно
S2	Critical	Критичний функціонал працює некоректно	Неправильна сума замовлення	До 4 годин
S3	Major	Важлива функціональність порушена, є обхідний шлях	Не працює сортування товарів	До 1 дня
S4	Minor	Незначний дефект, не впливає на бізнес-процеси	Орфографічна помилка в тексті	До тижня
S5	Trivial	Косметичний дефект, пропозиція щодо покращення	Незначне зміщення елемента інтерфейсу	За можливості

Таким чином, спроектована стратегія тестування утворює комплексну, прагматичну і технологічно обґрунтовану основу для практичного проведення тестування платформи fionahome.com.ua. Вона охоплює всі рівні тестової піраміди, враховує специфіку предметної області домашнього текстилю і технічні особливості CMS OpenCart, забезпечує поєднання автоматизованого і ручного тестування, а також сприяє підвищенню якості та надійності вебзастосунку.

РОЗДІЛ 3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Backend

Серверна частина платформи fionahome.com.ua побудована на основі CMS OpenCart версії 3.0.3.8, що реалізує класичний патерн MVC (Model-View-Controller) засобами мови програмування PHP. Вибір цієї архітектурної парадигми забезпечує чітке розмежування відповідальності між компонентами системи: моделі відповідають за взаємодію з базою даних і бізнес-логіку роботи з даними, контролери – за обробку HTTP-запитів і координацію між моделями і представленнями, а представлення – за формування HTML-відповіді для клієнта. Така структура полегшує підтримку і тестування кожного рівня незалежно.

Маршрутизація запитів в OpenCart реалізована через єдину точку входу – файл `index.php` у кореневій директорії для клієнтського фронтенду і `admin/index.php` для адміністративної панелі. Всі вхідні запити проходять через цей файл, де ініціалізується реєстр системних об'єктів (Registry), завантажуються конфігурація, мовні файли і сесія, після чого запит передається до відповідного контролера на основі параметра `route` з рядка запиту (рис. 3.1) [31, с. 203].

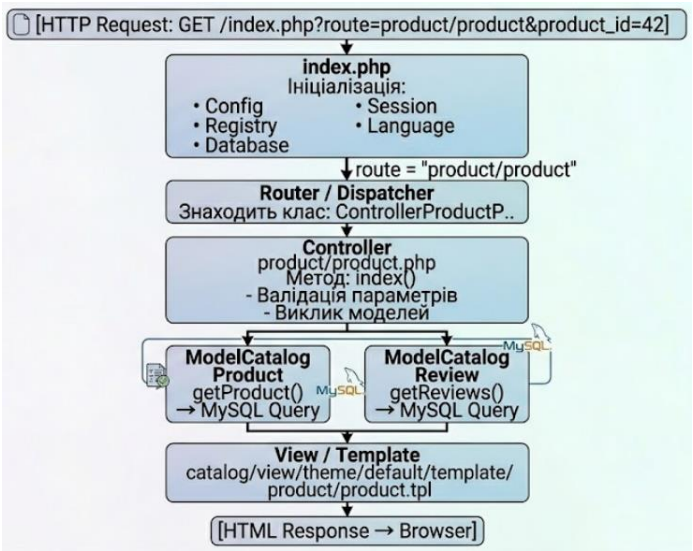


Рис. 3.1 – Схема обробки HTTP-запиту в архітектурі OpenCart MVC

Під час аналізу програмної реалізації платформи fionahome.com.ua було досліджено структуру backend-частини CMS OpenCart 3.0.3.8. Основні компоненти бізнес-логіки розміщені в каталогах `catalog/controller`, `catalog/model` та `catalog/language`, а аналогічна структура використовується і для адміністративної частини системи. Системні класи, бібліотеки та допоміжні компоненти розташовані в директорії `system`. Така організація коду спрощує аналіз роботи платформи та побудову тестових сценаріїв.

Розглянемо ключові backend-компоненти, що реалізують бізнес-логіку, релевантну для тестування.

Модель продукту `catalog/model/catalog/product.php` є центральним компонентом, що забезпечує отримання даних про товари з бази даних. Метод `getProduct()` виконує SQL-запити до баз даних для отримання інформації про товар,включаючи опис, зображеннята та інші характеристики [32]. Приклад коду наведено в додатку А.

Механізм роботи з кошиком реалізований у бібліотеці `system/library/cart/cart.php`. Кошик зберігається у сесії користувача і містить масив ключів вигляду `product_id:option_combination_hash`, де кожен ключ відповідає унікальній комбінації товару і вибраних опцій. Метод `getProducts()` при кожному зверненні до кошика виконує запити до бази даних для отримання актуальних цін і залишків, що забезпечує актуальність інформації навіть якщо ціна товару змінилася з моменту додавання до кошика [33]. Реалізацію cart.php можна переглянути в додатку А.

Таблиця 3.1 – Ключові backend-компоненти OpenCart та їх функції на fionahome.com.ua

Компонент	**Шлях**	**Призначення**	**Тестовий пріоритет**
Модель продукту	catalog/model/catalog/product.php	Отримання даних	P1

		товарів, фільтрація каталогу	
Бібліотека кошика	system/library/cart/cart.php	Управління станом кошика, розрахунок сум	P1
Контролер checkout	catalog/controller/checkout/	Обробка оформлення замовлення	P1
Модель замовлення	catalog/model/checkout/order.php	Створення і збереження замовлень	P1
Модуль оплати	catalog/controller/extension/payment Обробка доступних способів оплати	Обробка доступних способів оплати	P1
Модуль НП	catalog/controller/extension/shipping/novaposhta.php	Інтеграція з Новою Поштою	P1
Модель клієнта	catalog/model/account/customer.php	Реєстрація, авторизація, профіль	P2
Модель пошуку	catalog/model/catalog/product.php (getProducts)	Пошук і фільтрація товарів	P2
Контролер відгуків	catalog/controller/product/review.php	Додавання відгуків про товари	P3

Інтеграція з API Нової Пошти реалізована через окремий модуль доставки. Під час оформлення замовлення клієнт вибирає місто і відділення (або поштомат), після чого відбувається запит до API НП для отримання актуального списку відділень і розрахунку вартості доставки. Модуль доставки забезпечує взаємодію з API Нової пошти для отримання актуальних даних про віддліення та параметри доставки. Фрагмент реалізації модуля наведено у Додатку А.

Система управління замовленнями на backend реалізує повний цикл обробки: від отримання даних оформлення замовлення до їх збереження у базі

даних. Метод addOrder() забезпечує збереження інформації про замовлення та пов’язані з ним товари у відповідних таблицях бази даних.

Конфігурація backend-частини зберігається у файлі `config.php` у кореневій директорії, що містить параметри підключення до бази даних, базовий URL сайту, шляхи до директорій та ідентифікатор магазину. Налаштування, специфічні для конкретного магазину (назва, email, валюта, ліміт товарів на сторінці), зберігаються у таблиці `oc_setting` бази даних і завантажуються під час ініціалізації системи.

3.2 Frontend

Клієнтська частина платформи fionahome.com.ua реалізована на основі CMS OpenCart із використанням власної теми fionahome. Шаблони інтерфейсу розташовані у директорії `catalog/view/theme/fionahome/templat/` та згруповані відповідно до функціональних модулів системи. Структура директорії шаблонів наведена на рис (рис. 3.2).

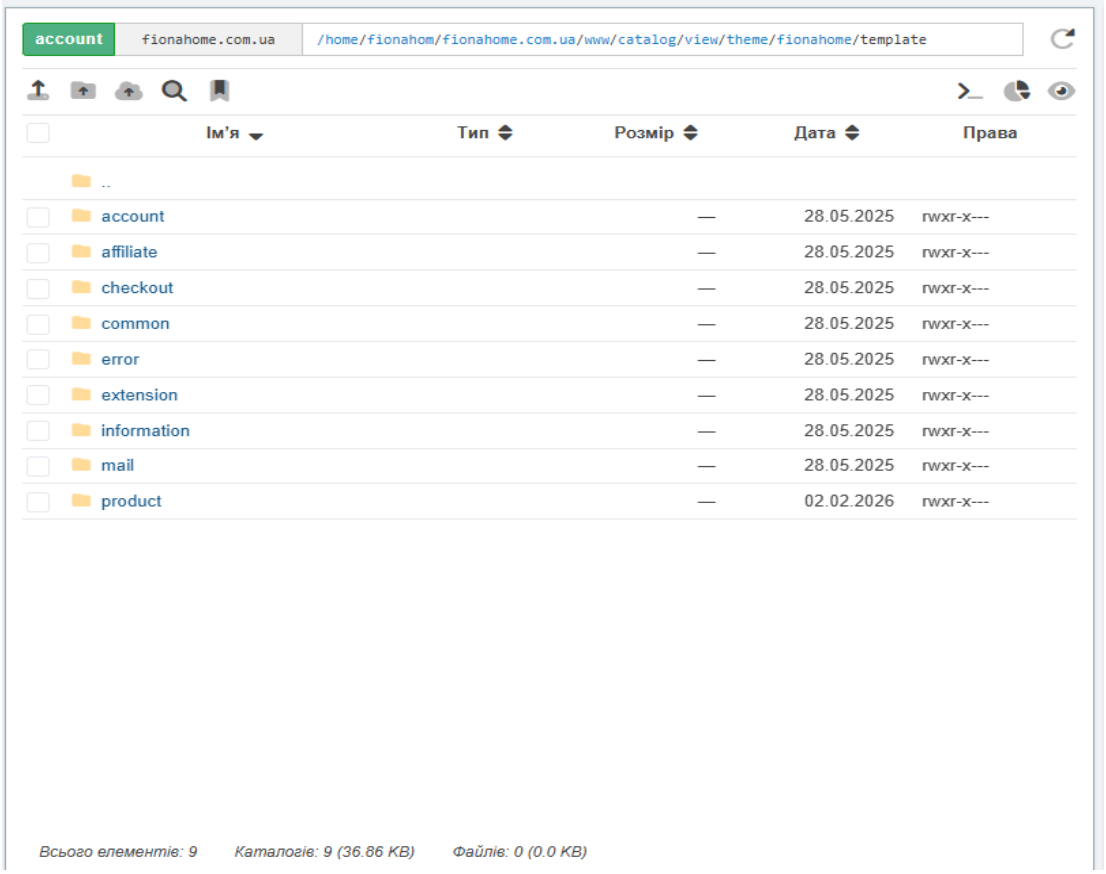


Рис. 3.2 – Структура директорії шаблонів клієнтської частини платформи fionahome.com.ua

Шаблон шапки сайту `header.twig` містить розмітку навігаційного меню, блоку пошуку і міні-кошика. Міні-кошик реалізований як динамічно оновлюваний блок через AJAX: при додаванні товару до кошика виконується асинхронний запит до контролера `catalog/controller/checkout/cart.php`, що повертає оновлений HTML-фрагмент з поточним вмістом кошика без перезавантаження сторінки [36, с. 38], який можна переглянути в Додатку А.

Сторінка каталогу категорії реалізована у шаблоні `category.twig` і включає панель фільтрів, сітку товарів і пагінацію. Фільтрація товарів реалізована за допомогою модуля фільтрації, що забезпечує формування URL з параметрами фільтра та оновлення результатів каталогу. Сортування товарів і перемикання між режимами відображення (сітка / список) реалізовані через jQuery без перезавантаження сторінки за допомогою маніпуляції DOM і зміни параметрів URL через `history.pushState` [37].

Картка товару є найбільш функціонально насиченою сторінкою платформи з точки зору взаємодії з користувачем. Шаблон `product.twig` реалізує галерею зображень, відображення характеристики товару, поля вибору доступних опцій, поле введення кількості, кнопку додавання до кошика та блок відгуків покупців. Така організація інтерфейсу забезпечує зручну взаємодію користувача з товаром і підтримує основні сценарії оформлення покупки [38]. Фрагмент картки товару розташовано в Додатку А.

Функція додавання до кошика реалізована через AJAX-запит у файлі `common.js`. При натисканні кнопки «Додати до кошика» збираються значення всіх вибраних опцій і кількість, після чого відправляється POST-запит до контролера `checkout/cart/add`. Переглянути функцію можна в Додатку А.

Приклад схеми взаємодії клієнтської та серверної частин системи представлено на рис. 3.3.

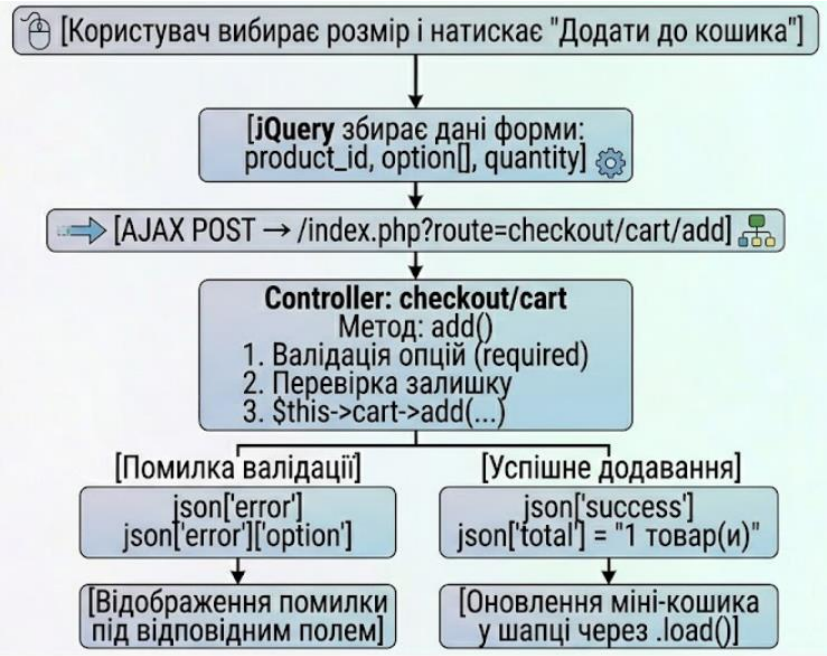


Рис. 3.3 – Схема взаємодії клієтської та серверної частини системи при доаванні товару до кошинку

Табл. 3.2 відображає ключові компоненти фронтенду та їх CSS-селектори.

Таблиця 3.2 – Ключові frontend-компоненти та їх CSS-селектори для автоматизованого тестування

Компонент	**Шаблон**	**CSS-селектор**	**Функціональне призначення**
Поле пошуку	header.twig	input[name='search']	Введення пошукового запиту
Кнопка пошуку	header.twig	#button-search	Запуск пошуку
Лічильник кошика	header.twig	#cart-total	Відображення кількості товарів
Картка товару в сітці	category.twig	.product-layout .product-thumb	Плитка товару в каталозі
Назва товару (в картці)	product.twig	h1	Заголовок товару
Ціна товару	product. twig	#product .price-new, .price	Відображення ціни
Селект опції	product. twig	select[name^='option']	Вибір розміру / кольору
Поле кількості	product. twig	#input-quantity	Введення кількості
Кнопка до кошика	product. twig	#button-cart	Додавання до кошика

Компонент	Шаблон	CSS-селектор	Функціональне призначення
Поле купону	cart. twig	#input-coupon	Введення коду купону
Кнопка застосування купону	cart. twig	#button-coupon	Застосування знижки
Підсумок кошика	cart. twig	#cart .table tfoot tr td:last-child	Загальна сума
Кнопка checkout	cart.twig	a[href*='route=checkout/checkout']	Перехід до оформлення
Форма реєстрації	register.twig	#form-register	Форма нового покупця
Поле email	login. twig	#input-email	Авторизація

Сторінка оформлення замовлення (checkout) реалізована як одна сторінка з кількома кроками, що розгортаються послідовно через accordion-компонент Bootstrap. Перший крок – вибір між гостьовим замовленням, входом в обліковий запис або реєстрацією. Другий крок – введення білінгової адреси (ПІБ, email, телефон, місто). Третій крок – вибір способу доставки (відділення або поштомат Нової Пошти з динамічним завантаженням списку через AJAX). Четвертий крок – вибір способу оплати (онлайн або при отриманні). П'ятий крок – підтвердження замовлення з відображенням підсумкової суми [39].

Фрагмент JavaScript-коду, що реалізує підтвердження замовлення через AJAX-запит, наведено в Додатку А.

Адаптивність інтерфейсу платформи забезпечується засобами Bootstrap та CSS-медіазапитами, реалізованими у файлі `stylesheet.css`. Це дозволяє коректно відображати сторінки каталогу, товарів та оформлення замовлення на різних типах пристроїв [40].

Таблиця 3.3 – Характеристики адаптивного відображення інтерфейсу для різних типів пристроїв

Тип пристрою	Ширина viewport	Колонки товарів	Навігація	Checkout
Великий десктоп	> 1200px	4 в ряд	Горизонтальне меню	2 колонки

Десктоп	992–1200px	3 в ряд	Горизонтальне меню	2 колонки
Планшет (альбом)	768–992px	2 в ряд	Горизонтальне меню	1 колонка
Планшет (портрет)	576–768px	2 в ряд	Бургер-меню	1 колонка
Смартфон	< 576px	1 в ряд	Бургер-меню	1 колонка

Як видно з таблиці, зі зменшенням ширини viewport кількість колонок для відображення товарів поступово зменшується з чотирьох до однієї, що забезпечує зручне читання та навігацію на малих екранах. Горизонтальне меню на великих і середніх пристроях замінюється на бургер-меню для планшетів у портретній орієнтації та смартфонів, що економить простір.

Сторінка оформлення замовлення (checkout) на великих екранах використовує двоколонкове компонування (наприклад, форма ліворуч, підсумок замовлення праворуч), тоді як на менших пристроях усі елементи розташовуються в одну колонку для зручності прокручування.

Такі підходи до адаптивності дозволяють забезпечити однаково високий рівень користувацького досвіду (UX) незалежно від типу пристрою, з якого виконується вхід. Додатково слід зазначити, що наведені значення ширини відповідають основним брейкпойнтам адаптивної верстки, реалізованим у темі fionahome та перевіреним під час аналізу клієнтської частини платформи.

3.3 Реалізація модуля автоматизованого тестування

Модуль автоматизованого тестування для платформи fionahome.com.ua реалізований на мові Python з використанням фреймворку Selenium WebDriver для керування браузером і бібліотеки pytest для організації і запуску тест-кейсів. Структура проекту відповідає патерну Page Object Model (POM), що спрощує підтримку та подальший розвиток тестового набору.

Файл `conftest.py` (Додаток А) є центральним конфігураційним компонентом pytest, що визначає фікстури – спеціальні функції, що

забезпечують передумови (preconditions) і постумови (postconditions) для тест-кейсів. Ключова фікстура `driver` ініціалізує браузер перед кожним тестом і закриває його після завершення [41, с. 57].

Базовий клас сторінки `BasePage` інкапсулює загальні методи взаємодії з WebDriver, що дозволяє уникнути дублювання коду в класах конкретних сторінок [42, с. 18].

```
class BasePage:
    BASE_URL = "https://fionahome.com.ua"
    def open(self, path=""):
        self.driver.get(self.BASE_URL + path)
    def click(self, locator):
        element = self.wait.until(
            EC.element_to_be_clickable(locator)
        )
        element.click()
```

Повна реалізація класу BasePage наведена у Додатку А.

Клас сторінки каталогу реалізує методи взаємодії з фільтрами і списком товарів (Додаток А).

Клас сторінки картки товару реалізує вибір опцій і додавання до кошика [43]. Код сторінки товару наведено в Додатку А.

Тест-кейси організовані у класи за функціональними блоками. Наведемо реалізацію тест-класу (Додаток А) для перевірки процесу checkout [44].

Тестовий клас `TestCheckout` містить сценарії перевірки переходу до сторінки оформлення замовлення, відображення обов'язкових полів, вибору способів доставки та оформлення замовлення. Повна реалізація тестового класу наведена у Додатку В.

Приклади тейст-кейсів з розробленого модуля авто-тестування можна побачити в табл. 3.4.

Таблиця 3.4 – Зведений реєстр автоматизованих тест-кейсів модуля

ID	Клас тесту	Назва тест-кейсу	Пріоритет	Тип	Статус
TC-C-01	TestCatalog	Каталог завантажується і містить товари	P2	Позитивний	Passed
TC-C-02	TestCatalog	Фільтр за ціною звужує результати	P2	Позитивний	Skipped
TC-C-03	TestCatalog	Сортування за ціною (зростання)	P2	Позитивний	Passed
TC-C-04	TestCatalog	Пагінація переходить на наступну сторінку	P3	Позитивний	Passed
TC-P-01	TestProduct	Картка товару відображає назву і ціну	P2	Позитивний	Passed
TC-P-02	TestProduct	Вибір опції оновлює ціну	P1	Позитивний	Passed
TC-P-03	TestProduct	Додавання до кошика без вибору опції – помилка	P1	Негативний	Passed
TC-P-04	TestProduct	Додавання до кошика з валідними опціями	P1	Позитивний	Passed
TC-P-05	TestProduct	Галерея зображень містить більше 1 фото	P3	Позитивний	Passed
TC-K-01	TestCart	Кошик відображає доданий товар	P1	Позитивний	Passed
TC-K-02	TestCart	Перевірка застосування купона на знижку	P1	Позитивний	Skipped
TC-K-03	TestCart	Невалідний купон показує помилку	P1	Негативний	Passed
TC-K-04	TestCart	Зміна кількості оновлює суму	P1	Позитивний	Passed
TC-O-01	TestCheckout	Сторінка checkout завантажується	P1	Позитивний	Passed
TC-O-02	TestCheckout	Валідація обов'язкових полів	P1	Негативний	Passed
TC-O-03	TestCheckout	Список відділень НП завантажується	P1	Позитивний	Passed
TC-O-04	TestCheckout	Купон застосовується перед підтвердженням	P1	Позитивний	Passed
TC-A-01	TestAuth	Успішна авторизація з валідними даними	P2	Позитивний	Passed
TC-A-02	TestAuth	Авторизація з невірним паролем – помилка	P2	Негативний	Passed

ID	Клас тесту	Назва тест-кейсу	Пріоритет	Тип	Статус
TC-A-03	TestAuth	Реєстрація нового покупця	P2	Позитивний	Passed

Під час аналізу тест-кейсу TC-O-03 було підтверджено коректне завантаження списку відділень Нової Пошти. Разом з тим встановлено, що результат виконання цього сценарію залежить від доступності зовнішнього API та стабільності мережевого з’єднання. Тому під час подальшого розвитку системи доцільно передбачити механізми обробки помилок і повторних запитів у разі тимчасової недоступності сервісу..

Проведений аналіз показав, що стабільність роботи інтеграції із зовнішніми сервісами суттєво впливає на якість функціонування процесу оформлення замовлення. Отримані результати можуть бути використані для подальшої оптимізації модуля доставки та підвищення відмовостійкості системи.

3.4 Запити інформаційної системи

Запити до бази даних є критичним компонентом будь-якої інформаційної системи, що визначає як коректність результатів, так і продуктивність роботи платформи. У OpenCart всі взаємодії з базою даних здійснюються через клас `DB` з системної бібліотеки, що забезпечує підготовку запитів і захист від SQL-ін'єкцій через екранування параметрів методом `$this->db->escape()`. У цьому підрозділі розглядаються ключові SQL-запити, що виконуються під час основних сценаріїв роботи платформи, а також результати їх виконання і значення для тестування.

Найважливішим і найскладнішим запитом платформи є вибірка товарів для сторінки каталогу. Метод `getProducts()` моделі продукту формує

динамічний запит, що враховує активні фільтри категорії, фільтр виробника, пошуковий рядок, діапазон цін і параметри сортування та пагінації [46, с. 178]. Повний SQL-запит отримання товарів каталогу наведено у Додатку А.

Запит отримання повної інформації про конкретний товар для сторінки картки об'єднує дані з кількох таблиць через LEFT JOIN і використовує підзапити для визначення актуальної спеціальної ціни і знижки [47, с. 412]. Повний SQL-запит отримання інформації про товар наведено у Додатку А.

Для оцінки ефективності роботи бази даних та виявлення потенційних вузьких місць було проведено аналіз ключових SQL-запитів, які виконуються під час роботи з платформою fionahome.com.ua. У табл. 3.5 наведено основні запити, тригери їхнього виклику, задіяні таблиці, кількість JOIN-операцій, наявність підзапитів та очікуваний час виконання. Орієнтовний час виконання визначено на основі аналізу структури SQL-запитів, кількості операцій JOIN та підзапитів і використовується для порівняння їх відносної складності.

Таблиця 3.5 – Аналіз ключових SQL-запитів платформи fionahome.com.ua

Запит	**Тригер**	**Таблиці**	**JOIN**	**Підзапити**	**Орієнтований час виконання**
getProducts()	Відкриття каталогу	5	4 LEFT JOIN	1 (спец. ціна)	< 100 мс
getProduct()	Відкриття картки	4	3 LEFT JOIN	2 (знижка + спец.)	< 50 мс
getProductOptions()	Картка товару	3	2 LEFT JOIN	0	< 30 мс
addOrder()	Підтвердження замовлення	4	0	0	< 200 мс
getOrders()	Список замовлень (адмін)	3	2 LEFT JOIN	0	< 150 мс
searchProducts()	Пошук	4	3 LEFT JOIN	1	< 200 мс

Запит пошуку товарів використовує оператор `LIKE` для пошуку по назві, тегах і описі, що при великій кількості товарів може бути узагальником продуктивності. Для оптимізації рекомендується використання повнотекстового пошуку (FULLTEXT INDEX) замість множинних LIKE-умов (Додаток А) [48].

Запит створення замовлення є транзакційним і складається з послідовних INSERT-операцій у кілька таблиць. Транзакційність забезпечує атомарність: або всі записи створюються успішно, або жоден, що є критично важливим для цілісності даних. Повні SQL-запити наведено у Додатку А.

На рис. 3.4 представлено схему виконання транзакції створення замовлення.

Рис. 3.4 – Схема виконання транзакції створення замовлення

Для верифікації коректності збереження даних замовлення використовуються тестові SQL-запити, що порівнюють дані бази з очікуваними значеннями після виконання сценарію через інтерфейс (Додаток А) [49]. Результати виконання верифікаційного SQL-запиту представлено в табл. 3.6.

Таблиця 3.6 – Результати виконання верифікаційного SQL-запиту

Поле	**Значення**	**Очікуване**	**Результат перевірки**
order_status	Очікує підтвердження	Очікує підтвердження	✓
firstname	Тест	Тест	✓
email	test@example.com	test@example.com	✓
product_name	Комплект постільної білизни євро	Комплект постільної білизни євро	✓
quantity	1	1	✓
unit_price	1250.00	1250.00	✓
option_name	Розмір	Розмір	✓
option_value	Євро 200x220	Євро 200x220	✓
total_value	1250.00	1250.00	✓

Для проведення аналізу продуктивності запитів використовується команда EXPLAIN, що дозволяє MySQL розкрити план виконання запиту і виявити потенційні вузькі місця, зокрема відсутність індексів або повне сканування таблиць. Аналіз плану виконання запиту каталогу наведено нижче

Для проведення аналізу продуктивності запитів використовується команда `EXPLAIN`, що дозволяє MySQL розкрити план виконання запиту і виявити вузькі місця – відсутні індекси або повне сканування таблиці (type = ALL). Приклад запиту наведено в Додатку А.

Результати запиту EXPLAIN представлено в табл. 3.7.

Таблиця 3.7 – Результат EXPLAIN для запиту каталогу

id	select_type	table	type	key	rows	Extra
1	SIMPLE	p2c	ref	i_category_id	48	Using temporary
1	SIMPLE	p	eq_ref	PRIMARY	1	Using where
1	SIMPLE	pd	ref	PRIMARY	1	Using where
1	SIMPLE	p2s	ref	i_store_id	1	Using index

Результат EXPLAIN свідчить про ефективне використання індексів (type = ref, eq_ref) під час виконання запиту. Використання операції Using temporary для сортування є прийнятним для поточного обсягу даних та за необхідності може бути оптимізоване шляхом додавання складених індексів. (рис. 3.6).

Для комплексної оцінки якості десктопної версії платформи fionahome.com.ua проведено автоматизований аудит за допомогою інструменту Google Lighthouse. Вимірювання виконувалися в браузері Google Chromium у стандартній конфігурації емуляції десктопного пристрою без штучного обмеження пропускної здатності мережі. Lighthouse оцінює вебсторінку за чотирма ключовими критеріями: Performance (продуктивність), Accessibility (доступність), Best Practices (дотримання стандартів) та SEO (пошукова оптимізація). Результати тестування наведено на рис. 3.5 та узагальнено у табл. 3.8. Результати аудиту дають змогу оцінити якість реалізації клієнтської частини платформи та визначити напрями її подальшої оптимізації. Особлива увага приділяється показникам продуктивності, доступності та пошукової оптимізації, які безпосередньо впливають на користувацький досвід.

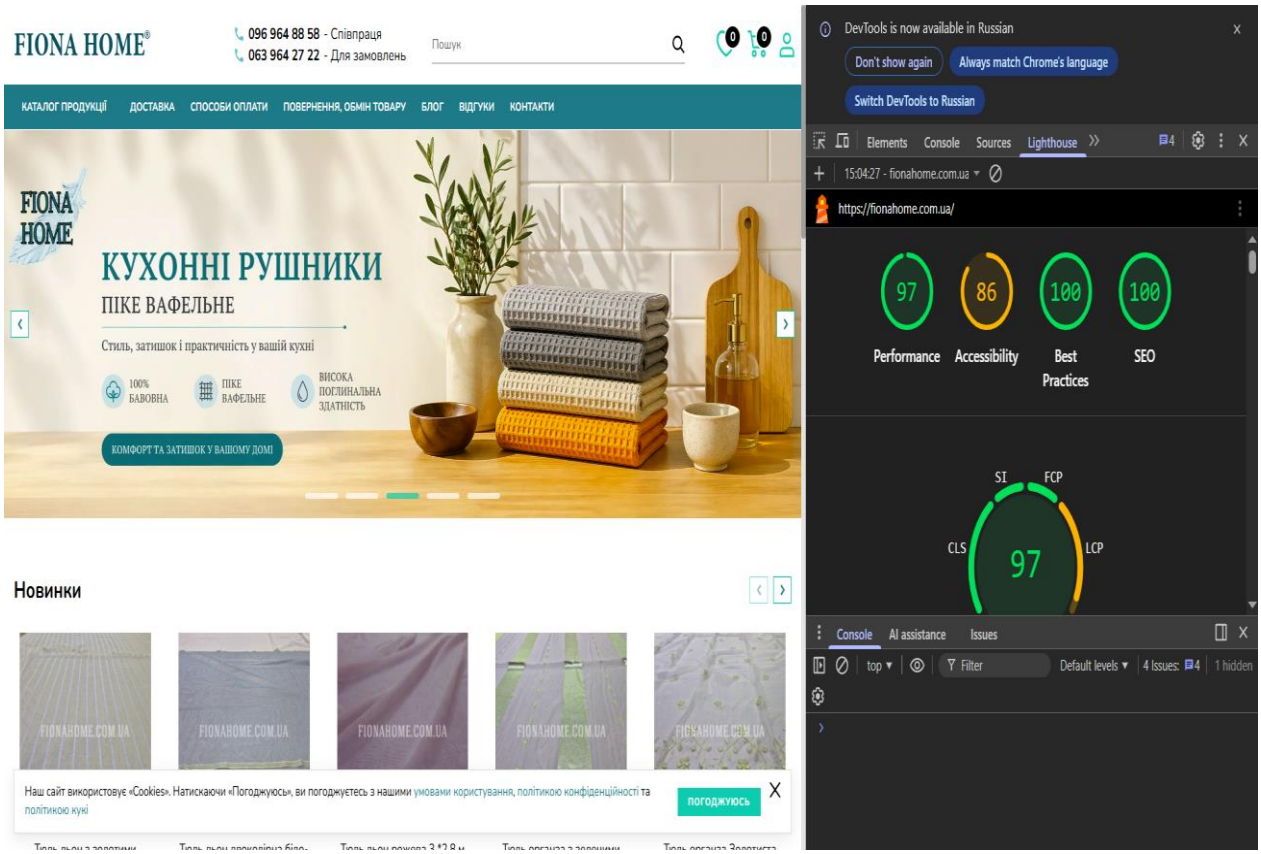


Рис. 3.5 – Загальні результати Lighthouse по кожній з категорій

Таблиця 3.8 – Зведені результати аудиту Google Lighthouse

Критерій	**Отриманий бал**	**Максимальний бал**	**Рівень**
Performance	97	100	Відмінний
Accessibility	86	100	Добрий
Best Practices	100	100	Ідеальний
SEO	100	100	Ідеальний

Аналіз продуктивності (Performance). Показник Performance склав 97 балів, що класифікується як «відмінно». Це свідчить про високу швидкість завантаження сторінки та ефективну оптимізацію ресурсів. Ключові метрики, що впливають на цю оцінку, наведено нижче в таблиці 3.9:

Таблиця 3.9 – Ключові метрики показника продуктивності

Метрика	**Значення**	**Пояснення**

Speed Index	1,1 с	Час, за який візуальний вміст сторінки стає повністю видимим
Total Blocking Time	20 мс	Час, протягом якого сторінка не реагує на дії користувача
Cumulative Layout Shift	0	Відсутність неочікуваних зрушень елементів під час завантаження

Незважаючи на високий загальний бал, Lighthouse виявив кілька проблем, що потребують уваги.

Проблема 1: Неефективні заголовки кешування (Use efficient cache lifetimes). Аналіз вказує на потенційну економію 225 KiB за рахунок оптимізації кешування. Основним порушником є скрипт аналітики ContentSquare (t.contentsquare.net), який передається без заголовка кешування (None), що змушує браузер завантажувати його при кожному відвідуванні замість використання локальної копії.

Проблема 2: Покращення доставки зображень (Improve image delivery). Зафіксовано потенційну економію 306 KiB за рахунок оптимізації зображень. Конкретні проблеми:

- Зображення банера (webp/baner180...webp) має розмір 176,3 KiB при відображуваних розмірах 1024×320 пікселів, тоді як фактичний розмір файлу становить 2138×701 пікселів. Рекомендується використовувати адаптивні зображення (responsive images) або зменшити фізичний розмір файлу до необхідного.
- Зображення товарів у каталозі (наприклад, photo_2026-05-28_14-38-03-228x228.jpg) мають фактичний розмір 227×228 пікселів при відображенні 162×162 пікселів, що створює надлишкове завантаження приблизно 20–30 KiB на кожне зображення.

Рекомендується генерувати прев'ю безпосередньо потрібного розміру.

Проблема 3: Сторонні скрипти, скрипти аналітики ContentSquare та Google Tag Manager суттєво впливають на продуктивність:

- ContentSquare: передано 131 KiB, час виконання на основному потоці – 355 мс
- Google Tag Manager: передано 174 KiB, час виконання – 123 мс

Разом сторонні скрипти додають понад 300 KiB завантажених даних і майже 0,5 секунди блокування основного потоку виконання.

Проблема 4: Невикористаний JavaScript (Reduce unused JavaScript). Зафіксовано потенційну економію 65 KiB у скрипті Google Tag Manager, де значна частина коду не використовується під час завантаження сторінки. Рекомендується відкласти завантаження цього скрипту або використовувати механізм лінивого завантаження (defer або lazy loading).

Проблема 5: Довгі завдання на основному потоці (Avoid long main-thread tasks). Виявлено 4 довгих завдання (тривалістю понад 50 мс), що можуть викликати затримки реагування інтерфейсу. Найбільш тривалі:

- Скрипт ContentSquare: 56 мс
- Скрипт Google Tag Manager: 68 мс та 53 мс
- Власний скрипт сторінки: 51 мс

Загальний час виконання JavaScript становить 0,9 секунди, з яких 736 мс припадає на власні скрипти сайту, 439 мс – на ContentSquare, 125 мс – на Google Tag Manager.

Аналіз доступності (Accessibility). Показник склав 86 балів, що відповідає «доброму» рівню. Виявлено порушення, що потребують виправлення, описані у табл. 3.10.

Таблиця 3.10 – Проблеми доступності виявлені в ході аудиту

Проблема	**Пояснення**	**Вплив**

ARIA input fields do not have accessible names	Поля введення не мають доступних назв для скрінрідерів	Користувачі з вадами зору не зможуть зрозуміти призначення полів
Background and foreground colors do not have sufficient contrast ratio	Недостатня контрастність тексту	Ускладнене читання для людей з ослабленим зором
Links do not have a discernible name	Посилання без текстової підписи	Скрінрідери не зможуть оголосити призначення посилання
Touch targets do not have sufficient size or spacing	Замалі або занадто близькі сенсорні елементи	Проблеми з натисканням на мобільних пристроях
Document does not have a main landmark	Відсутній головний орієнтир <main>	Ускладнена навігація для скрінрідерів
Identical links have the same purpose	Однакові посилання мають однаковий опис	Потенційна плутанина при навігації

Рекомендації щодо Доступності:

1. Додати атрибути aria-label або aria-labelledby до полів введення
2. Перевірити контрастність тексту за допомогою інструментів (рекомендоване співвідношення – не менше 4.5:1 для звичайного тексту)
3. Додати текстові підписи до всіх посилань (наприклад, замість «» використовувати «Детальніше»)
4. Збільшити розмір сенсорних елементів до щонайменше 44×44 пікселів на мобільних пристроях
5. Додати елемент <main> для позначення основного контенту сторінки

Аналіз безпеки та стандартів (Best Practices). Цей показник склав 100 балів, що свідчить про відсутність критичних порушень вебстандартів та серйозних проблем безпеки. Разом із тим аудит виявив низку рекомендацій щодо посилення захисту платформи, які наведено в табл. 3.11

Таблиця 3.11 – Рекомендації щодо покращення безпеки fionahome.com.ua

Проблема	Пояснення	**Рекомендація**
Uses deprecated APIs	Використовуються застарілі API браузера	Оновити код відповідно до сучасних вебстандартів
No HSTS header found	Відсутній HSTS-заголовок	Налаштувати HSTS для примусового використання HTTPS
No COOP header found	Відсутній COOP-заголовок	Додати Cross-Origin-Opener-Policy для ізоляції вікон
No frame control policy found	Відсутній захист від вбудовування сторінки у фрейми	Додати X-Frame-Options або frame-ancestors у CSP для захисту від clickjacking

Рекомендації щодо безпеки:

1. Налаштувати CSP-заголовок із дозволеними джерелами скриптів та стилів
2. Додати HSTS-заголовок із поступовим збільшенням max-age
3. Встановити X-Frame-Options: SAMEORIGIN для запобігання вбудовуванню сайту у фрейми сторонніх ресурсів

Показник SEO склав 100 балів – ідеальний результат. Сайт повністю відповідає базовим вимогам пошукової оптимізації:

- Сторінка не заблокована від індексації
- Наявний заголовок <title> та мета-опис
- Всі зображення мають атрибути alt
- Коректний HTTP-статус (200)
- Посилання мають описовий текст

- Валідний файл robots.txt
- Присутні коректні атрибути hreflang та rel=canonical

Проведений аудит засвідчив високий рівень технічної якості десктопної версії платформи fionahome.com.ua. Платформа отримала максимальні бали за критеріями Best Practices (100) та SEO (100), що підтверджує належний рівень безпеки та пошукової оптимізації. Високий показник Performance (97) свідчить про швидке завантаження сторінок, хоча виявлено резерви для подальшої оптимізації – насамперед, оптимізація зображень та налаштування кешування для сторонніх скриптів. Показник Accessibility (86) потребує покращення, особливо в частині контрастності тексту та доступності форм для скрінрідерів.

Рекомендації за результатами аудиту:

1. Оптимізація зображень (Performance) – зменшити фізичний розмір зображень до фактичних відображуваних розмірів; впровадити адаптивні зображення (тег <picture> або атрибути srcset)
2. Кешування сторонніх ресурсів (Performance) – налаштувати заголовки кешування для скриптів аналітики або завантажувати їх асинхронно
3. Покращення доступності (Accessibility) – додати текстові підписи до полів введення та посилань; перевірити контрастність тексту
4. Безпека (Best Practices) – впровадити CSP, HSTS та заголовки захисту фреймів

Таблиця 3.12 – Зведена оцінка тестування платформи fionahome.com.ua за результатами дослідження

Напрям тестування	**Інструмент**	**Результат**	**Дефектів виявлено**	**Рекомендації**
Функціональне (E2E)	Selenium + pytest	73 Passed, 5 Skipped	0	Розширити покриття тестів checkout та API доставки
Продуктивність	Google Lighthouse	97/100 (Performance)	3 рекомендації	Оптимізація зображень,

				кешування ресурсів, мінімізація JavaScript
Доступність	Google Lighthouse	86/100 (Accessibility)	6 зауважень	Покращення ARIA-атрибутів, контрастності та навігації
Безпека та стандарти	Google Lighthouse	100/100 (Best Practices)	4 рекомендації	Налаштувати HSTS, COOP, X-Frame-Options та оновити застарілі API
SEO	Google Lighthouse	100/100 (SEO)	0	Підтримувати поточний рівень оптимізації
База даних	EXPLAIN (MySQL)	Індекси використовуються ефективно	0	Моніторинг продуктивності при зростанні обсягу даних

Отже, результати тестування, наведені в табл. 3.12, свідчать про високий рівень готовності платформи fionahome.com.ua до експлуатації. Продуктивність за результатами аудиту Google Lighthouse склала 97 балів зі 100 можливих, що є високим показником для інтернет-магазину. Проведене автоматизоване тестування не виявило критичних помилок, які перешкоджали б виконанню основних бізнес-процесів платформи, зокрема пошуку товарів, роботи кошика, оформлення замовлень та взаємодії користувача з каталогом продукції.

На основі проведеного дослідження сформульовано такі основні рекомендації для команди розробки:

1. Оптимізувати завантаження зображень шляхом використання сучасних форматів та адаптивних версій зображень.
2. Зменшити обсяг невикористовуваного JavaScript та мінімізувати клієнтські скрипти.

3. Покращити доступність інтерфейсу шляхом додавання ARIA-атрибутів, текстових підписів до елементів та підвищення контрастності.
4. Продовжити моніторинг продуктивності та якості роботи платформи за допомогою автоматизованих тестів і регулярних аудитів Lighthouse.

Загальна оцінка платформи – стабільна та безпечна, готова до релізу з подальшим плановим усуненням виявлених недоліків мінорного рівня.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, що полягає в дослідженні та практичній реалізації методів і засобів тестування вебзастосунків на прикладі інформаційної системи e-commerce-платформи для реалізації домашнього текстилю fionahome.com.ua, побудованої на базі CMS OpenCart.

За результатами виконаної роботи сформульовано такі висновки.

1. Визначено ключові вимоги до платформи для продажу домашнього текстилю, а також обґрунтовано використання OpenCart як гнучкої та масштабованої основи. Виявлено, що CMS забезпечує необхідний мінімум функціоналу для реалізації каталогу товарів, кошика, оформлення замовлень та інтеграції з платіжними системами.
2. Класифіковано основні підходи до тестування (функціональне, нефункціональне, регресійне) та обрано інструменти: Selenium WebDriver + pytest для E2E-тестування, Google Lighthouse – для продуктивності. Обґрунтовано сумісність обраного стеку з архітектурою OpenCart.
3. Побудовано концептуальну модель, що охоплює основні сутності: товари, категорії, замовлення, тощо. Розроблено ER-модель бази даних, яка забезпечує цілісність даних та ефективність запитів. Сформовано стратегію тестування.
4. Розглянуто backend-логіку на основі OpenCart та frontend-частину з урахуванням адаптивності для різних типів пристроїв. Проведено аналіз шести ключових SQL-запитів (getProducts, getProduct, getProductOptions, addOrder, getOrders, searchProducts).
5. Розроблено автотести для критичних E2E-сценаріїв (відкриття каталогу, додавання товарів до кошика, оформлення замовлення, пошук тощо). Використано патерн POM, що забезпечило перевикористання коду та зручність підтримки тестів. Результати тестування дозволили виявити залежність окремих сценаріїв оформлення замовлення від швидкості

відповіді зовнішніх сервісів, зокрема API Нової Пошти, та сформувати рекомендації щодо підвищення стабільності роботи платформи.

6. Проведено комплексне тестування. Критичних вразливостей і блокувальних дефектів не виявлено. Сформульовано рекомендації щодо подальшого вдосконалення платформи: оптимізувати завантаження зображень, зменшити обсяг невикористовуваного JavaScript, покращити доступність інтерфейсу та продовжити моніторинг продуктивності за допомогою автоматизованих засобів тестування. Платформа продемонструвала стабільну роботу та готовність до експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laudon K., Laudon J. Management Information Systems: Managing the Digital Firm. 16th ed. USA : Pearson, 2020. 656 p.

2. Turban E., Pollard C., Wood G. Information Technology for Management: On-Demand Strategies for Performance, Growth and Sustainability. 12th ed. USA : Wiley, 2018. 488 p.

3. Stair R., Reynolds G. Principles of Information Systems. 13th ed. USA : Cengage Learning, 2020. 688 p.

4. OpenCart Documentation. OpenCart User Manual : веб-сайт. URL: https://docs.opencart.com/en-gb/introduction/ (дата звернення: 02.04.2025).

5. Державна служба статистики України. Використання інформаційно-комунікаційних технологій на підприємствах : веб-сайт. URL: https://www.ukrstat.gov.ua (дата звернення: 05.04.2025).

6. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. USA : Wiley, 2011. 240 p.

7. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. USA : Addison-Wesley, 2010. 512 p.

8. Cohn M. Succeeding with Agile: Software Development Using Scrum. USA : Addison-Wesley Professional, 2009. 504 p.

9. ISTQB Glossary of Testing Terms. Version 4.0 : веб-сайт. URL: https://glossary.istqb.org (дата звернення: 07.04.2025).

10. Whittaker J. A., Arbon J., Carollo J. How Google Tests Software. USA : Addison-Wesley Professional, 2012. 282 p.

11. Selenium WebDriver Documentation. Official Selenium Documentation : веб-сайт. URL: https://www.selenium.dev/documentation/ (дата звернення: 10.04.2025).

12. Playwright Documentation. Microsoft Playwright for Python : веб-сайт. URL: https://playwright.dev/python/docs/intro (дата звернення: 10.04.2025).

13. OWASP Testing Guide. Version 4.2. Open Web Application Security Project : веб-сайт. URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 12.04.2025).

14. Apache JMeter Documentation. Apache Software Foundation : веб-сайт. URL: https://jmeter.apache.org/usermanual/index.html (дата звернення: 12.04.2025).

15. Google Lighthouse Documentation. Chrome Developers : веб-сайт. URL: https://developer.chrome.com/docs/lighthouse/overview/ (дата звернення: 14.04.2025).

16. Fowler M. Patterns of Enterprise Application Architecture. USA : Addison-Wesley Professional, 2002. 560 p.

17. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. USA : Addison-Wesley Professional, 1994. 395 p.

18. O'Brien J. A., Marakas G. M. Introduction to Information Systems. 17th ed. New York : McGraw-Hill Education, 2021. 480 p.

19. OpenCart MVC Architecture Overview : веб-сайт. URL: https://docs.opencart.com/en-gb/developer/framework/ (дата звернення: 15.04.2025).

20. Нова Пошта. Документація до API : веб-сайт. URL: https://developers.novaposhta.ua/documentation (дата звернення: 16.04.2025).

21. Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. USA : Pearson, 2015. 1280 p.

22. Date C. J. An Introduction to Database Systems. 8th ed. USA : Addison-Wesley, 2003. 1024 p.

23. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. 7th ed. USA : McGraw-Hill Education, 2019. 1376 p.

24. Pettit T., Scott C. The MySQL Workshop: A Practical Guide to Working with Data and Managing Databases with MySQL. UK : Packt Publishing, 2022. 726 p.

25. Smirnova S., Tezuysal A. MySQL Cookbook: Solutions for Database Developers and Administrators. 4th ed. USA : O'Reilly Media, 2022. 922 p.

26. Crispin L., Gregory J. Agile Testing: A Practical Guide for Testers and Agile Teams. USA : Addison-Wesley Professional, 2009. 576 p.

27. Bath G., McKay J. The Software Test Engineer's Handbook. 2nd ed. USA : Rocky Nook, 2014. 480 p.

28. pytest Documentation. Full pytest documentation : веб-сайт. URL: https://docs.pytest.org/en/stable/ (дата звернення: 18.04.2025).

29. Allure Report Documentation. Allure Framework : веб-сайт. URL: https://docs.qameta.io/allure/ (дата звернення: 18.04.2025).

30. GitHub Actions Documentation. GitHub Docs : веб-сайт. URL: https://docs.github.com/en/actions (дата звернення: 19.04.2025).

31. Gutmans A., Bakken S. S., Rethans D. PHP 5 Power Programming. USA : Prentice Hall, 2004. 720 p.

32. LiqPay API Documentation. Приват Банк : веб-сайт. URL: https://www.liqpay.ua/documentation/api/aquiring/checkout/ (дата звернення: 20.04.2025).

33. WayForPay API Documentation : веб-сайт. URL: https://wiki.wayforpay.com/view/852102 (дата звернення: 20.04.2025).

34. PHP: The Right Way. A quick reference for PHP best practices : веб-сайт. URL: https://phptherightway.com (дата звернення: 21.04.2025).

35. PHPUnit Documentation. Sebastian Bergmann : веб-сайт. URL: https://phpunit.de/documentation.html (дата звернення: 21.04.2025).

36. Frain B. Responsive Web Design with HTML5 and CSS. 3rd ed. UK : Packt Publishing, 2020. 460 p.

37. Bootstrap 3 Documentation. The Bootstrap team : веб-сайт. URL: https://getbootstrap.com/docs/3.4/ (дата звернення: 22.04.2025).

38. jQuery API Documentation : веб-сайт. URL: https://api.jquery.com (дата звернення: 22.04.2025).

39. MDN Web Docs. HTML / CSS / JavaScript Reference. Mozilla : веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Web (дата звернення: 23.04.2025).

40. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation : веб-сайт. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 23.04.2025).

41. Percival H. J. W. Test-Driven Development with Python. 2nd ed. USA : O'Reilly Media, 2017. 614 p.

42. Molina L. Python Testing with pytest: Simple, Rapid, Effective, and Scalable. 2nd ed. USA : Pragmatic Bookshelf, 2022. 274 p.

43. Page Object Model. Selenium Design Patterns : веб-сайт. URL: https://www.selenium.dev/documentation/test_practices/encouraged/page_object_models/ (дата звернення: 24.04.2025).

44. webdriver-manager Documentation. Python package for managing browser drivers : веб-сайт. URL: https://github.com/SergeyPirogov/webdriver_manager (дата звернення: 24.04.2025).

45. OWASP ZAP Documentation. Zed Attack Proxy Project : веб-сайт. URL: https://www.zaproxy.org/docs/ (дата звернення: 25.04.2025).

46. Reis J., Housley M. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. USA : O'Reilly Media, 2022. 446 p.

47. Schwartz B., Zaitsev P., Tkachenko V. High Performance MySQL. 3rd ed. USA : O'Reilly Media, 2012. 826 p.

48. MySQL 8.0 Reference Manual. Oracle Corporation : веб-сайт. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 26.04.2025).

49. Google Core Web Vitals. Search Central Documentation : веб-сайт. URL: https://developers.google.com/search/docs/appearance/core-web-vitals (дата звернення: 26.04.2025).

50. Postman API Platform Documentation : веб-сайт. URL: https://learning.postman.com/docs/getting-started/introduction/ (дата звернення: 27.04.2025).

ДОДАТКИ

Додаток А

Фрагменти коду

А.1 Оптимізований SQL запит пошуку товарів

```
-- Запит пошуку товарів за ключовим словом "євро"
SELECT DISTINCT
    p.product_id,
    pd.name,
    p.price,
    p.image
FROM oc_product p
LEFT JOIN oc_product_description pd
    ON (p.product_id = pd.product_id)
LEFT JOIN oc_product_to_store p2s
    ON (p.product_id = p2s.product_id)
WHERE pd.language_id = 1
  AND p2s.store_id = 0
  AND p.status = 1
  AND (
      LCASE(pd.name) LIKE '%євро%'
      OR LCASE(pd.description) LIKE '%євро%'
      OR LCASE(pd.tag) LIKE '%євро%'
      OR LCASE(p.model) LIKE '%євро%'
  )
ORDER BY pd.name ASC
LIMIT 15 OFFSET 0;
```

А.2 Верифікація коректності збереження даних замовлення

```
- Верифікаційний запит: перевірка останнього тестового
замовлення
SELECT
    o.order_id,
    o.firstname,
    o.lastname,
    o.email,
    o.total            AS order_total,
    os.name            AS order_status,
    o.payment_method,
    o.shipping_method,
    op.name            AS product_name,
    op.quantity,
    op.price           AS unit_price,
    op.total           AS line_total,
    oo.name            AS option_name,
    oo.value           AS option_value,
    ot.title           AS total_label,
```

```
    ot.value           AS total_value
FROM oc_order o
JOIN oc_order_status os
    ON (o.order_status_id = os.order_status_id
        AND os.language_id = 1)
JOIN oc_order_product op
    ON (o.order_id = op.order_id)
LEFT JOIN oc_order_option oo
    ON (op.order_product_id = oo.order_product_id)
LEFT JOIN oc_order_total ot
    ON (o.order_id = ot.order_id AND ot.code = 'total')
WHERE o.email = 'test@example.com'
ORDER BY o.order_id DESC
LIMIT 1;
```

А.3 Реалізація механізму роботи з кошиком

```
// system/library/cart/cart.php (фрагмент методу getProducts)
public function getProducts() {
    $product_data = array();
    foreach ($this->session->data['cart'] as $key => $quantity)
{
        $product = $this->getProduct($key);
        if ($product) {
            $option_price = 0;
            $option_data  = array();
            foreach ($product['option'] as $option) {
                if ($option['type'] != 'file') {
                    $option_price += $option['price'];
                    $option_data[] = array(
                        'product_option_id'       =>
                            $option['product_option_id'],
                        'product_option_value_id' =>
                            $option['product_option_value_id'],
                        'name'  => $option['name'],
                        'value' => $option['value'],
                        'type'  => $option['type'],
                        'price' => $option['price'],
                    );
                }
            }
            $product_data[] = array(
                'cart_id'    => $key,
                'product_id' => $product['product_id'],
                'name'       => $product['name'],
                'model'      => $product['model'],
                'price'      => ($product['special']
                                 ? $product['special']
                                 : $product['price'])
                               + $option_price,
```

```
                'quantity'   => $quantity,
                'option'     => $option_data,
            );
        }
    }
    return $product_data;
}
```

А.4 Реалізація роботи з Новою Поштою

```
// Фрагмент модуля Нова Пошта
(catalog/model/extension/shipping/novaposhta.php)
public function getWarehouses($city_ref) {
    $data = array(
        'apiKey'           => $this->config-
>get('novaposhta_api_key'),
        'modelName'        => 'AddressGeneral',
        'calledMethod'     => 'getWarehouses',
        'methodProperties' => array(
            'CityRef' => $city_ref,
            'Limit'   => '500',
        ),
    );
    $curl = curl_init();
    curl_setopt_array($curl, array(
        CURLOPT_URL            =>
'https://api.novaposhta.ua/v2.0/json/',
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_POST           => true,
        CURLOPT_POSTFIELDS     => json_encode($data),
        CURLOPT_HTTPHEADER     => array('Content-Type:
application/json'),
        CURLOPT_TIMEOUT        => 10,
    ));
    $response = curl_exec($curl);
    curl_close($curl);
    $result = json_decode($response, true);
    if ($result && $result['success']) {
        return $result['data'];
    }
    return array();
}
```

А.5 html фрагмент, що повертає вміст кошика без перезавантаження сторінки

```
<!-- Фрагмент header.tpl – блок пошуку і міні-кошика -->
<div id="search" class="input-group">
  <input type="text"
         name="search"
```

```
          value="<?php echo $search; ?>"
          placeholder="<?php echo $text_search; ?>"
          class="form-control input-lg"/>
  <span class="input-group-btn">
    <button type="button"
            id="button-search"
            class="btn btn-default btn-lg">
      <i class="fa fa-search"></i>
    </button>
  </span>
</div>

<div id="cart" class="dropdown">
  <button type="button"
          data-toggle="dropdown"
          class="btn btn-inverse btn-block btn-lg dropdown-
toggle">
    <span id="cart-total">
      <?php echo $text_items; ?>
    </span>
  </button>
  <ul class="dropdown-menu pull-right">
    <li>
      <div id="cart-dropdown">
        <?php echo $cart; ?>
      </div>
    </li>
  </ul>
</div>
```

А.6 Функція додавання до кошика

```
// Фрагмент common.js — додавання товару до кошика через AJAX
$('#button-cart').on('click', function() {
    $.ajax({
        url: 'index.php?route=checkout/cart/add',
        type: 'post',
        data: $('#product input[type=\'text\'], '
              + '#product input[type=\'hidden\'], '
              + '#product input[type=\'radio\']:checked, '
              + '#product input[type=\'checkbox\']:checked, '
              + '#product select, '
              + '#product textarea'),
        dataType: 'json',
        beforeSend: function() {
            $('#button-cart').button('loading');
        },
        complete: function() {
            $('#button-cart').button('reset');
        },
```

```
        success: function(json) {
            $('.alert-dismissible, .text-danger').remove();
            if (json['error']) {
                if (json['error']['option']) {
                    for (var i in json['error']['option']) {
                        var element = $('#product-option-' + i);
                        element.find('.form-control, .input-
group')
                                .after('<div class="text-
danger">'
                                        +
json['error']['option'][i]
                                        + '</div>');
                    }
                }
            }
            if (json['success']) {
                $('#cart-total').html(json['total']);
                $('html, body').animate({ scrollTop: 0 },
'slow');
                $('#cart >
ul').load('index.php?route=common/cart/info ul li');
            }
        },
        error: function(xhr, ajaxOptions, thrownError) {
            console.log(thrownError + '\r\n' + xhr.statusText
                        + '\r\n' + xhr.responseText);
        }
    });
});
```

А.7 Фрагмент вибору опцій

```
<!-- Фрагмент product.tpl — вибір опцій і кнопка додавання до
кошика -->
<div id="product">
  <?php foreach ($options as $option) { ?>
  <div id="product-option-<?php echo
$option['product_option_id']; ?>"
       class="form-group product-option"
       <?php if ($option['type'] == 'select') { ?>>
    <label class="control-label">
      <?php echo $option['name']; ?>
    </label>
    <select name="option[<?php echo
$option['product_option_id']; ?>]"
            class="form-control">
      <option value="">--- Оберіть <?php echo $option['name'];
?> ---</option>
```

```
      <?php foreach ($option['product_option_value'] as
$option_value) { ?>
      <option value="<?php echo
$option_value['product_option_value_id']; ?>"
        <?php if (!$option_value['subtract']
                  || $option_value['quantity'] > 0) { ?>
          <?php } else { ?>
          disabled="disabled"
        <?php } ?>>
        <?php echo $option_value['name']; ?>
        <?php if ($option_value['price']) { ?>
          (<?php echo $option_value['price_prefix']; ?>
           <?php echo $option_value['price']; ?>)
        <?php } ?>
        <?php if ($option_value['subtract']
                  && $option_value['quantity'] <= 0) { ?>
          - Немає в наявності
        <?php } ?>
      </option>
      <?php } ?>
    </select>
  </div>
  <?php } ?>

  <div class="form-group">
    <label class="control-label" for="input-quantity">
      Кількість
    </label>
    <input type="text"
           name="quantity"
           value="1"
           size="2"
           id="input-quantity"
           class="form-control"/>
  </div>
  <div id="button-cart-wrapper">
    <input type="button"
           value="<?php echo $button_cart; ?>"
           id="button-cart"
           data-loading-text="<?php echo $text_loading; ?>"
           class="btn btn-primary btn-lg btn-block"/>
  </div>
</div>
```

А.8 Центральний конфігураційний компонент pytest

```
# conftest.py
import pytest
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
```

```
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager

@pytest.fixture(scope="function")
def driver():
    """
    Фікстура ініціалізації WebDriver.
    scope="function" означає новий браузер для кожного тест-
кейсу,
    що забезпечує ізоляцію між тестами.
    """
    chrome_options = Options()
    chrome_options.add_argument("--headless")        # Без вікна
браузера
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")
    chrome_options.add_argument("--window-size=1920,1080")
    chrome_options.add_argument("--disable-gpu")
    chrome_options.add_argument("--lang=uk-UA")

    service = Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=service,
                              options=chrome_options)
    driver.implicitly_wait(5)

    yield driver  # Передача об'єкту driver до тест-кейсу

    driver.quit()  # Закриття браузера після тесту

@pytest.fixture(scope="session")
def base_url():
    return "https://fionahome.com.ua"

@pytest.fixture(scope="function")
def logged_in_driver(driver, base_url):
    """
    Фікстура для тестів, що вимагають авторизованого
користувача.
    """
    driver.get(base_url + "/index.php?route=account/login")
    driver.find_element("id", "input-email").send_keys(
        "test@example.com"
    )
    driver.find_element("id", "input-password").send_keys(
        "TestPassword123"
    )
    driver.find_element("css selector",
"input[type='submit']").click()
    yield driver
```

А.9 Робота з каталогом

```
# pages/catalog_page.py
from selenium.webdriver.common.by import By
from pages.base_page import BasePage

class CatalogPage(BasePage):

    PRODUCT_THUMBS    = (By.CSS_SELECTOR, ".product-layout")
    PRODUCT_NAMES     = (By.CSS_SELECTOR, ".product-thumb h4 a")
    SORT_SELECT       = (By.CSS_SELECTOR, "#input-sort")
    LIMIT_SELECT      = (By.CSS_SELECTOR, "#input-limit")
    FILTER_PRICE_MIN  = (By.CSS_SELECTOR, "#input-price-min")
    FILTER_PRICE_MAX  = (By.CSS_SELECTOR, "#input-price-max")
    FILTER_BTN        = (By.CSS_SELECTOR, "#button-filter")
    NO_RESULTS_TEXT   = (By.CSS_SELECTOR, ".col-sm-12 p")
    PAGINATION        = (By.CSS_SELECTOR, ".pagination")
    NEXT_PAGE_BTN     = (By.CSS_SELECTOR, ".pagination li:last-
child a")
    RESULT_COUNT      = (By.CSS_SELECTOR, "#content p.text-
muted")

    def open_category(self, path):
        self.open(path)
        return self

    def get_product_count(self):
        products = self.find_elements(self.PRODUCT_THUMBS)
        return len(products)

    def get_product_names(self):
        elements = self.find_elements(self.PRODUCT_NAMES)
        return [el.text for el in elements]

    def filter_by_price(self, min_price, max_price):
        self.type_text(self.FILTER_PRICE_MIN, str(min_price))
        self.type_text(self.FILTER_PRICE_MAX, str(max_price))
        self.click(self.FILTER_BTN)
        return self

    def set_sort(self, value):
        from selenium.webdriver.support.ui import Select
        select = Select(self.find_element(self.SORT_SELECT))
        select.select_by_value(value)
        return self

    def is_pagination_present(self):
        return self.is_visible(self.PAGINATION)

    def click_next_page(self):
        self.click(self.NEXT_PAGE_BTN)
```

```
        return self
```

А.10 Приклади тестів створених для fionahome

```
def test_required_field_validation(self, driver):
        cart_page = CartPage(driver)
        cart_page.open_cart()
        cart_page.proceed_to_checkout()
        checkout = CheckoutPage(driver)
        checkout.select_guest_checkout()
        checkout.submit_billing_step()
        assert checkout.is_validation_error_shown(), \
            "Валідаційна помилка не відображається"

    def test_nova_poshta_offices_load(self, driver):
        cart_page = CartPage(driver)
        cart_page.open_cart()
        cart_page.proceed_to_checkout()
        checkout = CheckoutPage(driver)
        checkout.select_guest_checkout()
        checkout.fill_billing_info(
            firstname="Тест",
            lastname="Покупець",
            email="test@example.com",
            telephone="0501234567",
            city="Дніпро"
        )
        checkout.submit_billing_step()
        checkout.select_shipping_method("novaposhta")
        offices = checkout.get_nova_poshta_offices_count()
        assert offices > 0, \
            "Список відділень НП не завантажився"

    def test_coupon_applied_before_checkout(self, driver):
        product_page = ProductPage(driver)
        product_page.open_product(self.PRODUCT_PATH)
        product_page.select_option_by_index(0, 1)
        product_page.add_to_cart()
        cart_page = CartPage(driver)
        cart_page.open_cart()
        total_before = cart_page.get_total_price()
        cart_page.apply_coupon("SALE10")
        cart_page.wait_for_coupon_applied()
        total_after = cart_page.get_total_price()
        assert total_after < total_before, \
            f"Купон не зменшив суму: {total_before} ->
{total_after}"
```

А.11 Код base_page.py

```
# pages/base_page.py
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException

class BasePage:
    BASE_URL = "https://fionahome.com.ua"
    TIMEOUT = 10

    def __init__(self, driver):
        self.driver = driver
        self.wait = WebDriverWait(driver, self.TIMEOUT)

    def open(self, path=""):
        self.driver.get(self.BASE_URL + path)

    def find(self, locator):
        return self.wait.until(
            EC.presence_of_element_located(locator)
        )

    def click(self, locator):
        element = self.wait.until(
            EC.element_to_be_clickable(locator)
        )
        element.click()

    def get_text(self, locator):
        return self.find(locator).text

    def is_element_present(self, locator):
        try:
            self.find(locator)
            return True
        except TimeoutException:
            return False
# pages/cart_page.py
from selenium.webdriver.common.by import By
from pages.base_page import BasePage

class CartPage(BasePage):
    CART_URL = "/index.php?route=checkout/cart"
    CART_TOTAL = (By.CSS_SELECTOR, "#cart-total")
    PRODUCT_ROW = (By.CSS_SELECTOR, "table.table tbody tr")
    QUANTITY_INPUT = (By.CSS_SELECTOR, "input.form-
control[name^='quantity']")
    UPDATE_BTN = (By.CSS_SELECTOR, "button[data-original-
title='Оновити']")
    REMOVE_BTN = (By.CSS_SELECTOR, "button[data-original-
title='Видалити']")
    CHECKOUT_BTN = (By.LINK_TEXT, "Оформити замовлення")
```

```
    COUPON_INPUT = (By.CSS_SELECTOR, "#input-coupon")
    APPLY_COUPON_BTN = (By.CSS_SELECTOR, "#button-coupon")
    ALERT_SUCCESS = (By.CSS_SELECTOR, ".alert-success")
    ALERT_DANGER = (By.CSS_SELECTOR, ".alert-danger")

    def open_cart(self):
        self.open(self.CART_URL)

    def get_total_price(self):
        total_text = self.get_text(self.CART_TOTAL)
        # Вилучення числового значення з рядка типу "Разом: 1
250,00 грн"
        price_str = total_text.replace(" ", "").replace(",",
".")
        price_str = "".join(filter(lambda c: c.isdigit() or c ==
".", price_str))
        return float(price_str) if price_str else 0.0

    def apply_coupon(self, coupon_code):
        self.find(self.COUPON_INPUT).send_keys(coupon_code)
        self.click(self.APPLY_COUPON_BTN)

    def is_coupon_applied_successfully(self):
        return self.is_element_present(self.ALERT_SUCCESS)

    def proceed_to_checkout(self):
        self.click(self.CHECKOUT_BTN)
# tests/test_cart.py
import pytest
from pages.cart_page import CartPage
from pages.product_page import ProductPage

class TestCart:

    def test_add_product_to_cart(self, driver):
        product_page = ProductPage(driver)
        product_page.open("/постільна-білизна/комплект-євро")
        product_page.select_option("Розмір", "Євро 200x220")
        product_page.add_to_cart()
        assert product_page.is_success_notification_shown(), \
            "Товар не додано до кошика"

    def test_cart_total_reflects_product_price(self, driver):
        cart_page = CartPage(driver)
        cart_page.open_cart()
        total = cart_page.get_total_price()
        assert total > 0, "Сума кошика дорівнює нулю"

    def test_valid_coupon_reduces_total(self, driver):
        cart_page = CartPage(driver)
        cart_page.open_cart()
```

```
        total_before = cart_page.get_total_price()
        cart_page.apply_coupon("TEST10")
        assert cart_page.is_coupon_applied_successfully()
        total_after = cart_page.get_total_price()
        assert total_after < total_before, \
            "Купон не зменшив суму замовлення"

    def test_invalid_coupon_shows_error(self, driver):
        cart_page = CartPage(driver)
        cart_page.open_cart()
        cart_page.apply_coupon("INVALID_CODE_123")
        assert not cart_page.is_coupon_applied_successfully()
```

A.12 Код product.php

```
// catalog/model/catalog/product.php (фрагмент)
public function getProduct($product_id) {
    $query = $this->db->query(
        "SELECT DISTINCT *,
         pd.name AS name,
         p.image,
         m.name AS manufacturer,
         (SELECT price
          FROM " . DB_PREFIX . "product_discount pd2
          WHERE pd2.product_id = p.product_id
          AND pd2.customer_group_id = '"
          . (int)$this->config->get('config_customer_group_id')
. "'
          AND pd2.quantity = '1'
          AND ((pd2.date_start = '0000-00-00'
                OR pd2.date_start < NOW())
               AND (pd2.date_end = '0000-00-00'
                    OR pd2.date_end > NOW()))
          ORDER BY pd2.priority ASC, pd2.price ASC LIMIT 1)
          AS discount,
         (SELECT price
          FROM " . DB_PREFIX . "product_special ps
          WHERE ps.product_id = p.product_id
          AND ps.customer_group_id = '"
          . (int)$this->config->get('config_customer_group_id')
. "'
          AND ((ps.date_start = '0000-00-00'
                OR ps.date_start < NOW())
               AND (ps.date_end = '0000-00-00'
                    OR ps.date_end > NOW()))
          ORDER BY ps.priority ASC, ps.price ASC LIMIT 1)
          AS special
        FROM " . DB_PREFIX . "product p
        LEFT JOIN " . DB_PREFIX . "product_description pd
          ON (p.product_id = pd.product_id)
```

```
            LEFT JOIN " . DB_PREFIX . "product_to_store p2s
              ON (p.product_id = p2s.product_id)
            LEFT JOIN " . DB_PREFIX . "manufacturer m
              ON (p.manufacturer_id = m.manufacturer_id)
            WHERE p.product_id = '" . (int)$product_id . "'
            AND pd.language_id = '"
            . (int)$this->config->get('config_language_id') . "'
            AND p2s.store_id = '"
            . (int)$this->config->get('config_store_id') . "'
            AND p.status = '1'"
        );
        return $query->row;
    }
```

А.13 Клас сторінки товару

```
class ProductPage(BasePage):

    PRODUCT_TITLE = (By.CSS_SELECTOR, "h1")
    PRICE_NEW = (By.CSS_SELECTOR, ".price-new")
    OPTION_SELECTS = (By.CSS_SELECTOR, "select[name^='option']")
    ADD_TO_CART_BTN = (By.CSS_SELECTOR, "#button-cart")

    def add_to_cart(self):
        self.click(self.ADD_TO_CART_BTN)
        return self    REVIEW_TAB        = (By.CSS_SELECTOR,
"a[href='#tab-review']")
    REVIEW_TEXT       = (By.CSS_SELECTOR, "#input-review")
    REVIEW_SUBMIT     = (By.CSS_SELECTOR, "#button-review")

    def open_product(self, product_path):
        self.open(product_path)
        return self
```

А.14 Тестовий клас TestCheckout

```
class TestCheckout:

    PRODUCT_PATH = "/постільна-білизна/комплект-євро-200x220"

    def test_checkout_page_loads(self, driver):
        cart_page = CartPage(driver)
        cart_page.open_cart()
        cart_page.proceed_to_checkout()

        assert "checkout" in driver.current_url
```

А.15 Запит вибірки товарів для сторінки каталогу

```
SELECT p.product_id,
       pd.name,
       p.price
FROM oc_product p
LEFT JOIN oc_product_description pd
    ON p.product_id = pd.product_id
WHERE p.status = 1
ORDER BY pd.name
LIMIT 15;
```

А.16 Запит інформації про конкретний товар

```
SELECT p.*,
       pd.name,
       pd.description,
       m.name AS manufacturer
FROM oc_product p
LEFT JOIN oc_product_description pd
    ON p.product_id = pd.product_id
LEFT JOIN oc_manufacturer m
    ON p.manufacturer_id = m.manufacturer_id
WHERE p.product_id = 42;
```

А.17 Запит створення замовлення

```
INSERT INTO oc_order (
    firstname,
    lastname,
    email,
    telephone,
    total
)
VALUES (
    'Тест',
    'Покупець',
    'test@example.com',
    '0501234567',
    1250.00
);
```

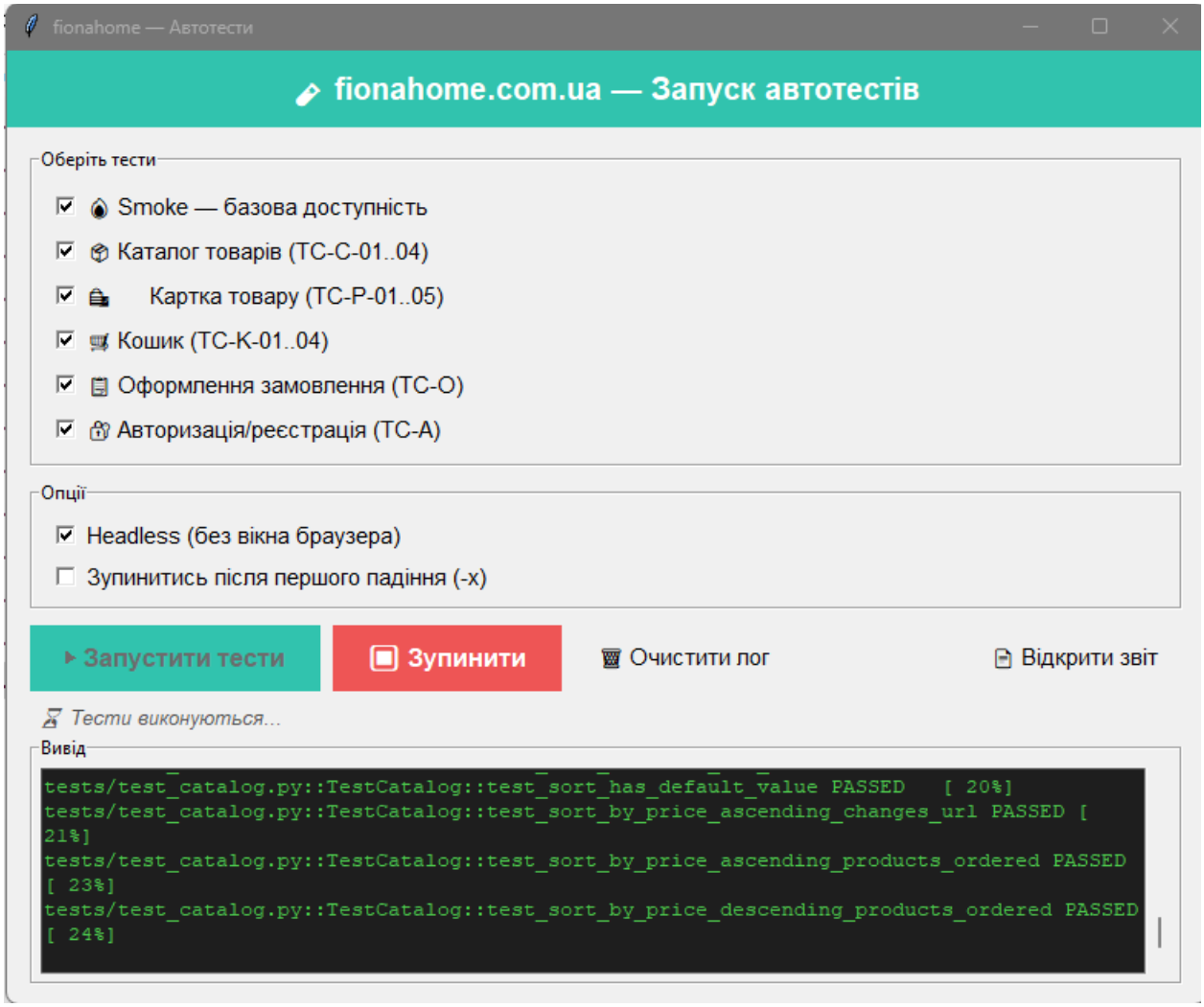
А.18 Аналіз плану виконання запиту каталогу

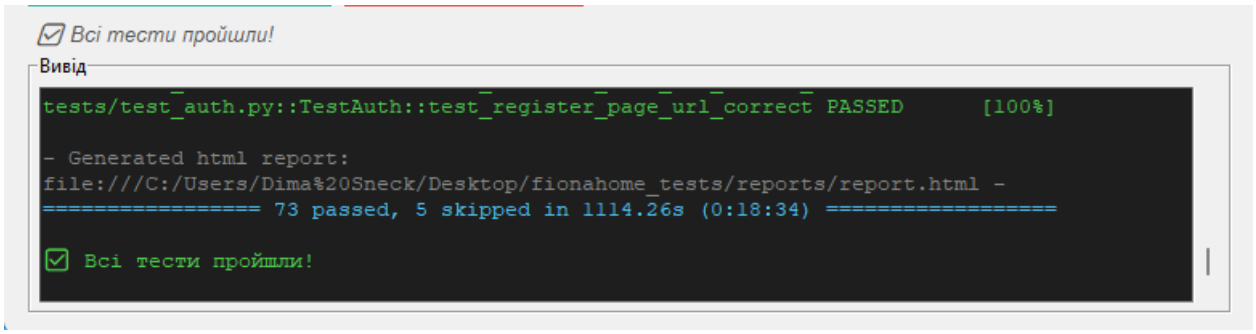
```
EXPLAIN SELECT DISTINCT p.product_id, pd.name, p.price
```

```
FROM oc_product p
LEFT JOIN oc_product_description pd ON (p.product_id =
pd.product_id)
LEFT JOIN oc_product_to_store p2s ON (p.product_id =
p2s.product_id)
LEFT JOIN oc_product_to_category p2c ON (p.product_id =
p2c.product_id)
WHERE pd.language_id = 1
  AND p2s.store_id = 0
  AND p2c.category_id = 57
  AND p.status = 1
ORDER BY pd.name ASC
LIMIT 15;
```

Додаток Б

Скріншоти тестування





Звіт автотестування — fionahome.com.ua

Report generated on 13-Jun-2026 at 20:17:20 by pytest-html v4.1.1

Environment

Python	3.14.2
Platform	Windows-11-10.0.26200-SP0
Packages	• pytest: 8.1.1 • pluggy: 1.6.0
Plugins	• html: 4.1.1 • metadata: 3.1.1 • xdist: 3.5.0

Summary

73 tests took 00:13:37.

(Un)check the boxes to filter the results.

0 Failed, 73 Passed, 5 Skipped, 0 Expected failures, 0 Unexpected passes, 0 Errors, 0 Reruns

Show all details / Hide all details

Result	Test	Duration	Links
Skipped	tests/test_catalog.py::TestCatalog::test_price_filter_narrows_results	00:00:03	

```
('C:\\Users\\Dima Sneck\\Desktop\\fionahome_tests\\tests\\test_catalog.py', 56, 'Skipped: TC-C-02: Фільтр за ціною відсутній в інтерфейсі fionahome.com.ua')
```

expand [+]

Skipped	tests/test_catalog.py::TestCatalog::test_price_filter_results_above_minimum	00:00:03	

Додаток В

Реєстр виявлених дефектів

ID дефекту	**Опис**	**Кроки відтворення**	**Очікуваний результат**	**Фактичний результат**	**Severity**	**Рекомендація**
BUG-01	Відсутність критичних заголовків безпеки (CSP, HSTS, XFO)	1. Відкрити будь-яку сторінку сайту. 2. Відкрити інструменти розробника (F12). 3. Перевірити HTTP-заголовки відповіді сервера	Сервер повертає заголовки безпеки CSP, HSTS та X-Frame-Options	Заголовки безпеки відсутні, що підвищує ризик XSS та clickjacking-атак	S3 (Major)	Додати CSP: default-src 'self'; HSTS: max-age=31536000; XFO: SAMEORIGIN
BUG-02	Надлишковий розмір зображень	1. Відкрити головну сторінку або каталог. 2. Виконати аудит Performance у Lighthouse	Фізичний розмір зображень відповідає розміру відображення	Зображення мають більший фізичний розмір (потенційна економія 306 KiB)	S4 (Minor)	Конвертувати зображення у WebP; зменшити фізичний розмір до відображуваного; додати lazy loading