

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ
КНИЖОК

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-513сп

Спеціальності

123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ТІЩЕНКО В.О.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТЯГУНОВ Д.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

« » 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГІЩЕНКО Вадим Олексійович

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи інтернет-магазину книжок

керівник проєкту (роботи) к.т.н., доцент, СКРУПСЬКИЙ С.Ю.

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «14» квітня 2026 року № 160

2. Строк подання студентом проєкту (роботи) 16.05.2026 р.

3. Вихідні дані до проєкту (роботи) веб додаток інтернет-магазину книжкової продукції «СНІТАНКА», серверна мова програмування PHP, клієнтська мова програмування JavaScript, реляційна СУБД MySQL, мова гіпертекстової розмітки HTML5, каскадні таблиці стилів CSS3, фронтед-фреймворк Bootstrap5.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області та порівняння аналогів;

2) Проєктування системи;

3) Розробка програмного забезпечення системи;

4) Випробування системи та аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	СКРУПСЬКИЙ С.Ю.		
Нормоконтроль	ГОЛУБ Т.В		

7. Дата видачі завдання «07» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Отримання завдання	до 07.04.2026	
2	Проведення огляду предметної області	до 10.04.2026	
3	Аналіз предметної області та порівняння аналогів	до 19.04.2026	
4	Формування вимог до системи	до 24.04.2026	
5	Проектування системи	до 28.04.2026	
6	Проектування бази даних	до 01.05.2026	
7	Проектування інтерфейсу користувача	до 05.05.2026	
8	Розробка програмного забезпечення системи	до 10.05.2026	
9	Випробування системи та аналіз результатів	до 11.05.2026	
10	Проектування інтерфейсу користувача	до 12.05.2026	
11	Оформлення пояснювальної записки	до 16.05.2026	
12	Проходження нормоконтролю	до 29.05.2026	
13	Перевірка на наявність академічного плагіату	до 30.05.2026	
14	Проходження рецензування	до 31.05.2026	

Студент(ка)

(підпис)

Вадим ТИЩЕНКО

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Степан СКРУПСЬКИЙ

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
100 с., 2 табл., 33 рис., 28 джерел.

БАЗА ДАНИХ, ВЕБДОДАТОК, ЕЛЕКТРОННА КОМЕРЦІЯ, ІНТЕРНЕТ-МАГАЗИН, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, BOOTSTRAP5, JAVASCRIPT, MYSQL, PHP.

Об'єкт розробки – процес розробки комп'ютерної системи інтернет-магазину книжкової продукції «СНІТАНКА».

Мета роботи – підвищення ефективності процесу продажу книжкової продукції через мережу Інтернет шляхом скорочення часу оформлення замовлень та оптимізації використання ресурсів серверного обладнання.

Робота складається з чотирьох розділів.

Перший розділ присвячено аналізу предметної області. Проведено огляд сучасного стану книжкового ринку України та тенденцій розвитку сектору e-commerce, зроблено порівняльний аналіз існуючих аналогів, на основі якого зроблено вимоги до розроблюваної системи.

У другому розділі розкрито питання проектування системи. Обґрунтовано вибір трирівневої клієнт-серверної архітектури. За допомогою мови UML розроблено моделі поведінки системи, які формалізують процеси асинхронної взаємодії та спроектовано логічну і фізичну структуру реляційної бази даних, та створено DBML-опис.

Третій розділ присвячено розробці програмного забезпечення. Описано процес створення адаптивної клієнтської частини з використанням HTML5, CSS3 та фреймворку Bootstrap 5. На базі Web Storage API було реалізовано інтерактивний кошик. За допомогою JavaScript зроблено механізм динамічної фільтрації товарів. Серверну частину реалізовано мовою PHP: написано скрипти

для безпечної автентифікації користувачів, обробки асинхронних запитів та транзакційної взаємодії з СУБД MySQL.

У четвертому розділі наведено результати випробувань системи. Проведено тестування коректності відображення адаптивного інтерфейсу на різних пристроях. Виконано перевірку клієнтської логіки, надійності механізмів валідації введення даних та зроблен аудит серверної обробки транзакцій та збереження каскадної цілісності в базі при оформленні замовлень.

В результаті виконання дипломної роботи створено повноцінну розподілену комп'ютерну систему інтернет-магазину, яка успішно вирішує завдання підвищення ефективності онлайн-продажів шляхом оптимізації швидкодії та раціонального розподілу обчислювального навантаження. Програмний продукт є повністю працездатним та готовим до практичного впровадження у реальні бізнес-процеси підприємств книготоргівлі.

ЗМІСТ

Вступ.....	7
Скорочення та умовні позначки	9
1 Аналіз предметної області та порівняння аналогів	10
1.1 Аналіз ринку книжкових інтернет-магазинів	10
1.2 Огляд та порівняння існуючих аналогів	11
1.3 Формування вимог до системи.....	14
1.3.1 Функціональні вимоги	14
1.3.2 Нефункціональні вимоги	17
1.4 Обґрунтування вибору технологій	18
1.4.1 HTML5	19
1.4.2 CSS3	20
1.4.3 Bootstrap 5	21
1.4.4 JavaScript	22
1.4.5 PHP	23
1.4.6 MySQL	24
1.5 Висновки до розділу 1	24
2 Проектування системи	25
2.1 Архітектура програмного забезпечення	25
2.1.1 Рівні архітектурної моделі вебдодатка.....	26
2.1.2 Механізм взаємодії компонентів та життєвий цикл запиту	27
2.2 Проектування бази даних	29
2.3 Проектування інтерфейсу користувача.....	34
2.4 Моделювання поведінки системи	37
2.5 Обґрунтування апаратних вимог до комп'ютерної системи	42
2.6 Висновок до розділу 2	44
3 Розробка програмного забезпечення системи	45
3.1 Розробка клієнтської частини та каталогу	45

3.2.1 Реалізація локального збереження даних за допомогою Web Storage API	50
3.2.2 Логіка управління станом кошика та функції додавання і видалення товарів	52
3.2.3 Динамічне оновлення інтерфейсу в реальному часі.....	54
3.2.4 Робота з Offcanvas-компонентом та подіями.....	56
3.3 Реалізація підсистеми авторизації та ідентифікації	57
3.4 Розробка модуля оформлення замовлення	61
3.5 Висновок до розділу 3	65
4 Випробування системи та аналіз результатів	66
4.1 Випробування інтерфейсу користувача	66
4.2 Перевірка механізмів навігації та фільтрації товарів	69
4.3 Перевірка роботи кошика та динамічного перерахунку	72
4.4 Валідація даних при оформленні замовлення	74
4.5 Перевірка серверної логіки та цілісності бази даних	76
4.6 Висновок до розділу 4	79
Висновок.....	80
Перелік джерел посилання	82
Додаток А Лістинги програм.....	85
Додаток В Слайди презентації	94

ВСТУП

Значна частина торгівлі книжковою продукцією зараз відбувається через інтернет. При цьому більшість покупців використовують для замовлень мобільні пристрої. Аналіз наявних книжкових інтернет-магазинів показує, що багато з них побудовані на базі важких універсальних CMS-систем або перевантажені зайвими графічними елементами та рекламою. Це призводить до повільного завантаження сторінок та створює зайве навантаження на сервери.

Для невеликих книготорговельних підприємств використання таких громіздких рішень часто є економічно та технічно недоцільним. Виникає потреба у створенні легкої, спеціалізованої комп'ютерної системи, яка виконуватиме лише необхідний набір функцій: зручний пошук, швидке додавання в кошик та безпечне оформлення замовлення. Актуальність розробки полягає в тому, що проектування системи з нуля з використанням сучасних підходів дозволяє зробити її значно швидшою та дешевшою в обслуговуванні порівняно з готовими шаблонними конструкторами[1].

Розробка комп'ютерної системи інтернет-магазину переходить до створення двох взаємопов'язаних частин: клієнтської та серверної. Проблема багатьох старих систем полягає в синхронній обробці запитів, коли при кожній дії користувача (наприклад, зміні кількості книг у кошику) сторінка повністю перезавантажується. Завдання полягає в тому, щоб усунути цей недолік, застосувавши архітектурний підхід, де інтерфейс працює реактивно (через JavaScript), а сервер лише приймає кінцеві дані для запису в базу (через PHP).

Підставою для розробки є необхідність автоматизації процесів обліку та продажу книжок в режимі онлайн. Вихідними даними є набір обраних технологій: реляційна база даних MySQL для надійного зберігання інформації, мова PHP для реалізації бізнес-логіки на сервері, а також JavaScript і фреймворк Bootstrap 5 для створення адаптивного клієнтського інтерфейсу.

Значна увага буде приділяється створенню адаптивного інтерфейсу користувача, реалізації інтерактивного кошика замовлень, системи авторизації та безпечної обробки даних. Також важливим аспектом є забезпечення швидкої взаємодії між клієнтською та серверною частиною за допомогою асинхронної передачі даних та організація надійного збереження інформації у базі даних.

Отже, дипломна робота спрямована на реалізацію сучасного інтернет-магазину книжкової продукції, здатного забезпечити ефективну, швидку та зручну взаємодію користувача з вебсистемою, а також відповідати сучасним вимогам електронної комерції та веброзробки.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	– База даних
ДРБ	– дипломна робота бакалавра
ІТ	– інформаційні технології
СУБД	– система управління базами даних
ТЗ	– технічне завдання
API	– інтерфейс програмування додатків (Application Programming Interface)
CSS	– каскадні таблиці стилів (Cascading Style Sheets)
HTML	– мова розмітки гіпертексту (HyperText Markup Language)
HTTP	– протокол передавання гіпертексту (HyperText Transfer Protocol)
IDE	– інтегроване середовище розробки (Integrated Development Environment)
JS	– об'єктно-орієнтована мова програмування (JavaScript)
JSON	– текстовий формат обміну даними (JavaScript Object Notation)
PHP	– скриптова мова програмування загального призначення (Hypertext Preprocessor)
UI	– користувацький інтерфейс (User Interface)
UML	– уніфікована мова моделювання (Unified Modeling Language)
UX	– користувацький досвід (User Experience)
XXS	– міжсайтовий скриптинг (Cross-Site Scripting)

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОРІВНЯННЯ АНАЛОГІВ

1.1 Аналіз ринку книжкових інтернет-магазинів

Сучасний стан книжкового ринку України характеризується глибокою трансформацією, спричиненою як глобальними процесами цифровізації, так і специфічними внутрішніми факторами. Перехід книготоргівлі в онлайн-площину став не просто додатковим каналом продажу, а критично важливою умовою виживання та розвитку галузі в умовах нестабільності [1].

Динаміка розвитку сектору e-commerce у 2026 році свідчить про стабільне зростання попиту на книжкову продукцію, що інтегрована у категорію «Хобі та дозвілля». Попри економічну виваженість споживачів, обсяги реалізації товарів через інтернет-майданчики продовжують зростати. Основними чинниками стабілізації ринку виступають його висока адаптивність до кризових явищ та інтенсивна розробка нових цифрових сервісів, спрямованих на задоволення потреб сучасного читача [2].

Для детального розуміння середовища, у якому функціонують суб'єкти книготоргівлі, доцільно структурувати ринок за типами гравців та їхніми ключовими характеристиками (табл. 1.1):

Таблиця 1.1 – Структура ринку книготоргівлі

Сегмент ринку	Представники	Ключові особливості та динаміка
Універсальні маркетплейси	Yakaboo, Rozetka	Займають домінуючу частку ринку. Характеризуються величезним асортиментом, потужною логістикою та розвиненими цифровими екосистемами (e-books, audio).
Спеціалізовані мережі	Книгарня «Є», Буква	Демонструють успішну омніканальну модель. Акцент на сервісі, атмосфері та бренді;

Продовження таблиці 1.1

Сегмент ринку	Представники	Ключові особливості та динаміка
Видавничі магазини (D2C)	КСД, Vivat, ВСЛ	Прямі продажі від виробника до споживача. Мають найвищу динаміку зростання завдяки ексклюзивним цінам, програмам лояльності та контролю над якістю контенту.
Соціальна комерція	Instagram/TikTok крамниці	Нішевий сегмент, що швидко зростає серед молоді. Орієнтовані на візуальний маркетинг, лідерів думок та імпульсивні покупки.

Основним драйвером розвитку ринку у 2026–2026 роках є мобільна комерція (m-commerce). Частка замовлень, зроблених через смартфони, перевищила 60%, що вимагає від розробників переходу до стратегії Mobile-first. Ще одним вагомим трендом є персоналізація на основі штучного інтелекту: впровадження алгоритмів розумних рекомендацій та AI-чатботів стає обов'язковим стандартом для конкурентоспроможного веб ресурсу [2].

Варто також відзначити зміну споживчої поведінки. Український читач став більш раціональним, що проявляється у зростанні попиту на електронні формати та активному використанні систем порівняння цін. Таким чином, ринок перебуває у фазі технологічного оновлення, де виграють проєкти, здатні запропонувати максимальну швидкість взаємодії та персоналізований підхід до кожного клієнта [3].

1.2 Огляд та порівняння існуючих аналогів

Для обґрунтування розробки нового програмного продукту було проведено дослідження основних гравців на ринку України, серед яких ключовими є Yakaboo, Книгарня «Є» та Клуб Сімейного Дозвілля. Кожна з цих платформ має

свою специфічну стратегію, яку важливо врахувати при проектуванні власної системи.

Найбільшим конкурентом з точки зору технологій є маркетплейс Yakaboo (рис.1.1). Його бізнес-модель орієнтована на максимальне охоплення аудиторії через величезний асортимент та інтеграцію цифрового контенту. Плюсом цієї платформи є наявність розвиненого мобільного застосунку та власної екосистеми для читання електронних книг, що робить її лідером у технічному плані. Проте з погляду інтерфейсу сайт має суттєвий мінус – він занадто перевантажений категоріями та рекламними банерами. Це створює труднощі з юзабіліті, оскільки користувачу важко сфокусуватися на виборі книги через інформаційний шум [4]. Для мого проекту це є фактором, що інтерфейс має бути лаконічнішим.

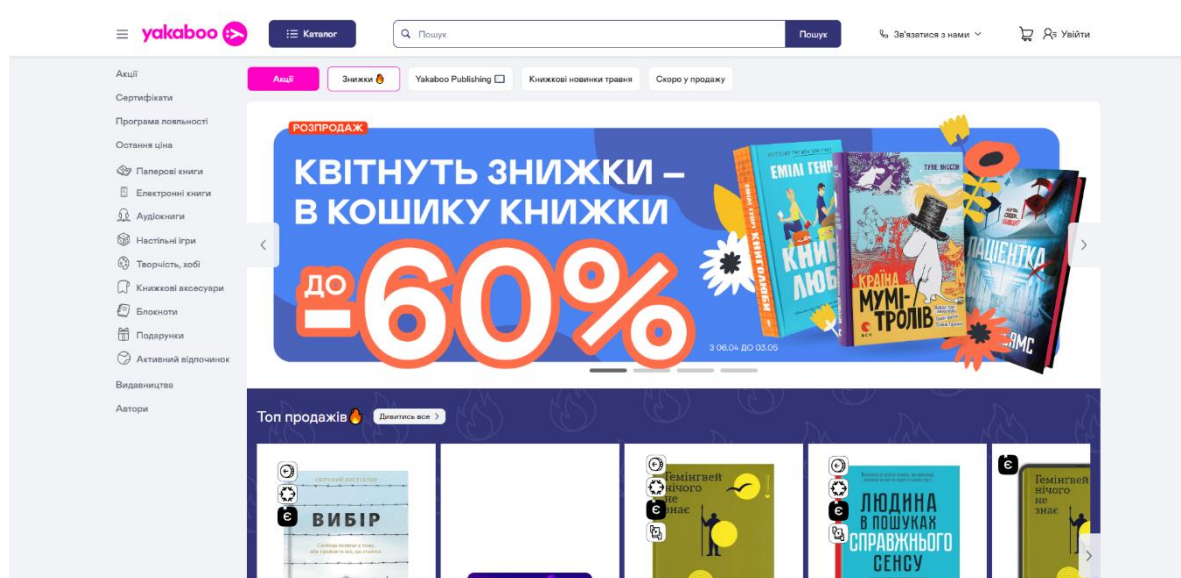


Рисунок 1.1 – Приклад інтернет-магазину книг «Yakaboo»

Книгарня «Є» (рис.1.2) демонструє зовсім інший підхід, фокусуючись на бренді та лояльності спільноти. Головна перевага їхньої онлайн-платформи полягає у зручній синхронізації з фізичними точками видачі, що забезпечує клієнту комфортний сервіс [5]. Сайт має досить логічну архітектуру, де акцент зроблено на зручному пошуку за видавництвами. Однак мінусом є певна консервативність: компанія майже не розвиває сегмент електронних книг і додаткових інтерактивних сервісів для читачів. Це створює нішу для розробки

більш гнучкого продукту, який поєднуватиме затишний дизайн із сучасними мультимедійними можливостями.

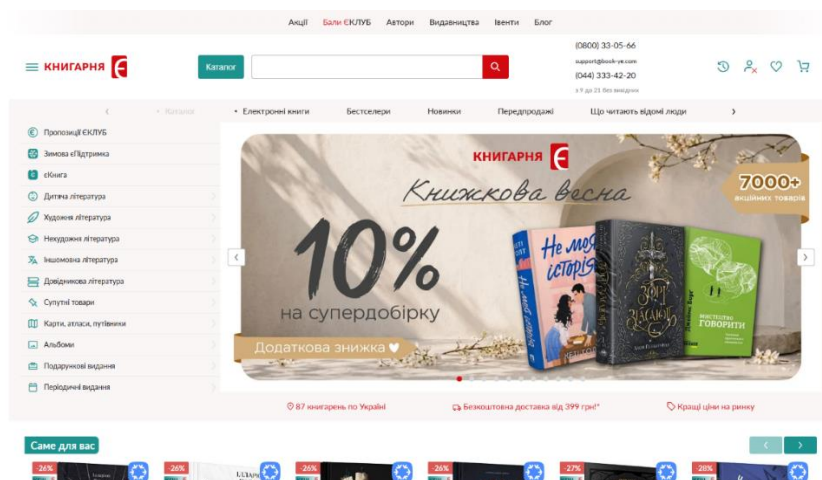


Рисунок 1.2 – Приклад інтернет-магазину книг Книгарня «Є»

Окремо варто виділити Клуб Сімейного Дозвілля (рис.1.3)., який тримає позиції завдяки власному видавництву та низьким цінам [6]. Плюс цієї платформи у її унікальному контенті та величезній базі постійних клієнтів, які залучені через клубну систему. Проте технічна реалізація сайту є найбільш слабким місцем цього конкурента. Інтерфейс виглядає дещо застарілим, а система обов'язкових замовлень для отримання знижок часто відлякує нових користувачів. Також відсутність сучасних алгоритмів рекомендацій робить процес підбору літератури менш персоналізованим, ніж того вимагає сучасний ринок.

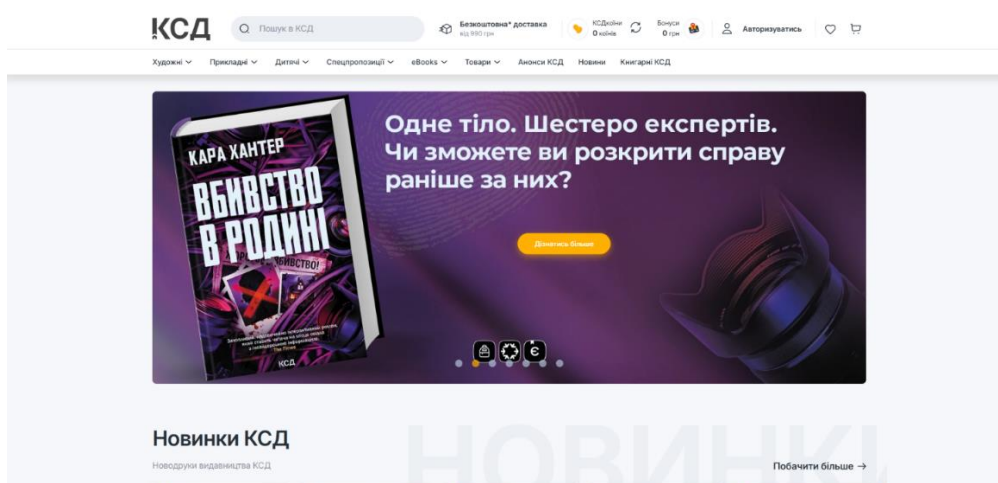


Рисунок 1.3 – Приклад інтернет-магазину книг «Клуб Сімейного Дозвілля»

Підсумовуючи цей аналіз, можна зробити висновок, що на ринку існує запит на інтернет-книгарню, яка була б такою ж швидкою та функціональною, як Yakaboo, але при цьому залишалася простою та зручною у використанні, як Книгарня «Є». У моїй роботі увага буде приділена розробці інтуїтивно зрозумілого веб-додатка, де акцентом стане реалізація інтерактивного кошика, системи авторизації та процесу оформлення замовлення із миттєвим збереженням даних у реляційну базу [7], що дозволить уникнути технічних та UX-недоліків, виявлених у чинних лідерів ринку.

1.3 Формування вимог до системи

Розробка будь-якого програмного продукту, зокрема системи електронної комерції, вимагає чіткого визначення вимог до його роботи [8]. Для інтернет-магазину книжкової продукції ці вимоги формуються на основі попередньо проведеного аналізу ринку та виявлених недоліків існуючих аналогів. Розроблювана система повинна забезпечувати можливість повноцінної, швидкої та безпечної взаємодії користувача з веб додатком на всіх етапах: починаючи від першого ознайомлення з каталогом і закінчуючи успішним оформленням покупки. Для цього треба визначити функціональні та нефункціональні вимоги, яких треба дотримуватись для розробки системи

1.3.1 Функціональні вимоги

Функціональні вимоги визначають базову поведінку розроблюваної системи електронної комерції, перелік її можливостей та процес взаємодії з користувачем на всіх етапах його перебування на сайті [9]. Для інтернет-магазину книжкової продукції ці вимоги формують ядро бізнес-логіки та описують конкретні дії, які програмний продукт повинен виконувати у відповідь на запити клієнта або системні події.

Для забезпечення повноцінного циклу купівлі-продажу, функціонал системи логічно розподіляється на декілька ключових підсистем (модулів), кожна з яких має чітко регламентовані вимоги.

Вимоги до модуля каталогу та навігації. Підсистема каталогу є основною вітриною магазину, тому до неї висуваються наступні функціональні вимоги:

- система повинна автоматично генерувати та виводити структурований асортимент книжкової продукції у вигляді інформативних товарних карток;
- кожна товарна картка має відображати базову інформацію про видання (назву, автора, ціну, зображення обкладинки) та містити активні елементи взаємодії, такі як кнопка додавання товару до кошика;
- програмний продукт повинен забезпечувати механізм глибокої категоризації літератури;
- система має надавати користувачеві інструменти динамічної фільтрації для миттєвого сортування та відображення книг виключно за обраним жанром чи тематикою.

Вимоги до підсистеми інтерактивного кошика. Кошик є центральним елементом для накопичення замовлення, до якого пред'являються жорсткі вимоги щодо інтерактивності:

- система повинна забезпечувати збереження обраних користувачем товарів локально, використовуючи механізми пам'яті браузера;
- кошик має автоматично і в режимі реального часу перераховувати загальну вартість замовлення та оновлювати бейдж кількості товарів при кожному додаванні або видаленні позиції;
- стан кошика повинен безперервно зберігатися та коректно відтворюватися при навігації користувача між різними сторінками інтернет-магазину без втрати обраних даних;
- доступ до інформації про поточне замовлення має надаватися швидко, без необхідності обов'язкового переходу на окрему веб сторінку до моменту безпосереднього оформлення покупки [10].

Вимоги до модуля ідентифікації та авторизації користувачів. Оскільки електронна комерція передбачає обробку персональних даних, система повинна мати надійний механізм контролю доступу:

- система повинна надавати функціонал реєстрації нових клієнтів та авторизації існуючих користувачів за їхніми обліковими даними;
- після успішної авторизації серверна частина повинна ініціювати сеанс користувача та управляти ним за допомогою Cookie-файлів;
- система зобов'язана дозволяти перегляд каталогу доступний усім відвідувачам, проте функціонал оформлення замовлення повинен блокуватися для неідентифікованих клієнтів і бути доступним виключно для авторизованих користувачів;

ідентифікація клієнта є обов'язковою вимогою для можливості реалізації майбутніх програм лояльності та надання персональних знижок конкретним покупцям.

Вимоги до підсистеми оформлення замовлення. Процес замовлення покупки є критичним етапом, який вимагає високої надійності обробки даних:

- система повинна забезпечувати первинну перевірку даних, введених у форму оформлення замовлення, на стороні клієнта (за допомогою JavaScript та вбудованого Forms API стандарту HTML5) для запобігання відправці порожнього кошика чи некоректних контактних даних;
- передача зібраної інформації на сервер має здійснюватися асинхронно за допомогою технології AJAX у форматі JSON-пакета;
- серверна частина повинна забезпечувати повторну, жорстку валідацію отриманих даних, перевіряти наявність активного сеансу авторизованого користувача та відфільтровувати вхідні змінні;
- система має безпечно записувати дані про нове замовлення у реляційну базу даних MySQL, гарантуючи чітку прив'язку загальної інформації про покупку до профілю клієнта у таблиці orders [11];

- після створення основного запису та отримання його унікального ідентифікатора, бекенд системи повинен здійснювати циклічний запис кожної окремої купленої книги у пов'язану таблицю деталей замовлення `order_items`;
- після успішного збереження транзакції у базі даних, система повинна повернути клієнту підтвердження, автоматично очистити локальне сховище кошика та перенаправити користувача або вивести відповідне сповіщення про успіх операції [12].

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають атрибути якості розроблюваної системи, такі як продуктивність, надійність, безпека та зручність використання [13]. Для інтернет-магазину книжкової продукції ці вимоги описують не конкретні дії, а те, як саме система повинна працювати, щоб забезпечити конкурентоспроможний рівень користувацького досвіду та стабільність технічної інфраструктури.

Вимоги до швидкодії та продуктивності. У сфері електронної комерції швидкість роботи веб додатка є критичним фактором утримання клієнтів. До розроблюваної системи висуваються наступні вимоги щодо продуктивності:

- система повинна характеризуватися високою швидкістю та максимальною оптимізацією клієнт-серверної взаємодії;
- програмний продукт має миттєво реагувати на дії користувача, не перериваючи його роботу очікуванням повного перезавантаження веб сторінки;
- для зменшення навантаження на сервер та пришвидшення відгуку інтерфейсу, зберігання проміжних даних має відбуватися виключно на стороні клієнта у локальній пам'яті браузера;
- мінімізація загального обсягу завантажуваних скриптів повинна досягатися шляхом використання чистого JavaScript без підключення важких сторонніх бібліотек, таких як jQuery [14].

Вимоги до ергономіки та адаптивності інтерфейсу. Враховуючи сучасні тенденції зростання мобільної комерції, інтерфейс користувача повинен відповідати найвищим стандартам використання:

- інтерфейс магазину повинен коректно відображатися, зберігати пропорції та зручність навігації на будь-яких типах пристроїв, включаючи десктопні монітори, планшетні комп'ютери та смартфони;

- система має підтримувати кросбраузерність, тобто однаково стабільно працювати та відображатися у всіх сучасних веб браузерях;

- дизайн інтерфейсу повинен бути лаконічним та інтуїтивно зрозумілим, щоб користувач міг сфокусуватися на виборі літератури без перевантаження інформаційним шумом.

Вимоги до надійності та безпеки даних. Оскільки система обробляє персональні дані покупців та фіксує деталі фінансових транзакцій, вимоги до безпеки є стандартними:

- серверна частина системи повинна забезпечувати жорстку фільтрацію всіх вхідних змінних, що надходять від клієнта у форматі JSON, для превентивного захисту бази даних від SQL-ін'єкцій [15];

- управління сесіями авторизованих користувачів має здійснюватися безпечно за допомогою механізму Cookie-файлів, виключаючи можливість несанкціонованого доступу до чужих облікових записів;

- система повинна мати клієнтську валідацію форм введення, що блокуватиме спроби відправки на сервер порожніх або некоректних даних;

- реляційна база даних MySQL повинна гарантувати цілісність збереженої інформації за рахунок використання зовнішніх ключів, які жорстко пов'язують кожне замовлення з профілем конкретного покупця та унеможливають появу неідентифікованих транзакцій [16].

1.4 Обґрунтування вибору технологій

Для реалізації розроблених вимог та забезпечення високої продуктивності інтернет-магазину, було проведено вибірку інструментів розробки. Вибір

та забезпечує миттєвий перерахунок загальної вартості та кількості товарів у реальному часі. HTML5 має потужний вбудований Forms API, що підтримує спеціалізовані типи полів введення (наприклад, `type="email"`, `type="tel"`) та атрибути обов'язковості і шаблонів (`required`, `pattern`). Це дозволяє реалізувати первинну жорстку перевірку даних (наприклад, перевірку формату номера телефону, адреси електронної пошти або захист від відправки форми з порожніми полями) безпосередньо на стороні браузера. Такі можливості допоможуть у створенні майбутніх форм заповнення даних [17, с. 576-583].

1.4.2 CSS3

Для візуального оформлення та реалізації користувацького інтерфейсу (UI/UX) було обрано стандарт каскадних таблиць стилів CSS3 (рис. 1.5).



Рисунок 1.5 – Приклад стандарту стилів CSS3

Ця технологія дозволяє повністю відділити структуру сторінки (HTML) від її зовнішнього вигляду, що забезпечує чистоту коду, спрощує його подальшу підтримку та масштабування. Використання CSS3 є головним для створення сучасного, лаконічного та інтуїтивно зрозумілого дизайну інтернет-магазину, що безпосередньо впливає на враження користувачів та допомагає уникнути недоліків перевантажених інтерфейсів, виявлених під час аналізу конкурентів. Вибір CSS3 також пов'язаний із використанням обраного фреймворку Bootstrap 5. Сучасні модулі CSS3, такі як Flexbox та CSS Grid, є технічною основою адаптивної сітки даного фреймворку. Їх застосування дозволить суттєво пришвидшити процес верстки товарного каталогу, забезпечивши гнучке та

симетричне вирівнювання елементів інтерфейсу незалежно від розмірів вікна браузера [17, с. 430-432].

1.4.3 Bootstrap 5

Для забезпечення кросбраузерності, адаптивності та пришвидшення процесу розробки клієнтської частини інтернет-магазину було обрано фреймворк Bootstrap 5 (рис. 1.6).



Рисунок 1.6 – Приклад фреймворка Bootstrap 5

Використання готового CSS/JS-фреймворку є загальноприйнятим стандартом у сучасній веб-розробці, оскільки це дозволяє уникнути написання великих обсягів. Ключовою архітектурною особливістю Bootstrap 5 є повна відмова від залежності від бібліотеки jQuery. Усі інтерактивні компоненти фреймворку (модальні вікна, випадаючі списки, панелі) переписані на чистому (Vanilla) JavaScript. Це суттєво зменшує загальний обсяг завантажуваних скриптів (зменшення "ваги" сторінки), пришвидшує час реакції інтерфейсу та оптимізує продуктивність веб додатка, що є критично важливим для утримання користувачів у сфері електронної комерції. Згідно з функціональними вимогами, магазин повинен мати зручний кошик замовлень, який не перекриває основний контент і не вимагає переходу на окрему сторінку до моменту оформлення покупки. Bootstrap 5 пропонує вбудований компонент Offcanvas (виїзна бічна панель), яка підходить для цього завдання. Завдяки цьому компоненту буде реалізовано швидкий доступ користувача до списку обраних товарів, їх кількості

та загальної суми з будь-якої сторінки каталогу. Сітка Bootstrap 5 базується на CSS Flexbox і дозволяє швидко створювати складні макети, які автоматично підлаштовуються під розміри екрана (від смартфонів до над широких моніторів) [18].

1.4.4 JavaScript

Для забезпечення динамічної поведінки веб додатка та реалізації складної клієнтської логіки було обрано мову програмування JavaScript (рис. 1.7).



Рисунок 1.7 – Приклад мови програмування JavaScript

У контексті розробки інтернет-магазину використання виключно серверних технологій для обробки кожної дії користувача є неефективним, оскільки призводить до постійних перезавантажень сторінки. JavaScript вирішує цю проблему, переносячи значну частину обчислень та логіки взаємодії на сторону клієнта (браузера користувача). Найважливішою функцією JavaScript у розробленій системі є забезпечення фонові комунікації з сервером. При оформленні замовлення використовується сучасний інтерфейс Fetch API для асинхронної відправки зібраних даних (інформації з форми та масиву товарів) у форматі JSON до серверного обробника. Це дозволяє здійснити запис транзакції в базу даних непомітно для користувача, без перезавантаження сторінки, що покращує користувацький досвід і відповідає стандартам сучасного e-commerce сегмента. Магазин також повинен мати інтерактивний кошик, здатний зберігати інформацію між сесіями. Саме за допомогою методів JavaScript для

роботи з localStorage організовано надійне збереження обраних книг у пам'яті браузера. Завдяки парсингу та серіалізації даних (через об'єкт JSON), система здатна оперувати структурами даних без необхідності створювати тимчасові таблиці в базі даних для неавторизованих користувачів [17, с. 331-350].

1.4.5 PHP

Для реалізації серверної частини (back-end) інтернет-магазину та управління ключовими бізнес-процесами було обрано мову програмування PHP (рис. 1.8).



Рисунок 1.8 – Приклад мови програмування PHP

Оскільки клієнтська частина відповідає лише за відображення та тимчасове зберігання даних, необхідний надійний інструмент для їх остаточної валідації, збереження та безпечної обробки на стороні сервера. Вибір PHP як базової технології бекенду обґрунтований її високою ефективністю у вирішенні специфічних завдань розробленої систем. Проєкт вимагає надійної та швидкої взаємодії з обраною системою управління базами даних MySQL. PHP надає вбудовані інтерфейси для взаємодії з СУБД. У межах розробки це дозволить реалізувати транзакційну логіку модуля оформлення замовлень, безпечну вставку даних про клієнта та загальну суму у таблицю orders, миттєве отримання унікального ідентифікатора створеного замовлення та подальший циклічний запис масиву куплених книг до пов'язаної таблиці order_items [17, с. 66-94].

1.4.6 MySQL

Для організації зберігання, структурування та обробки даних інтернет-магазину було обрано систему управління реляційними базами даних (СУБД) MySQL. Електронна комерція вимагає точності у роботі з фінансами, замовленнями та клієнтськими профілями. MySQL є однією з найпопулярніших та найстабільніших СУБД у світі, яка ідеально підходить для вирішення завдань розробленого веб додатка. Бізнес-логіка інтернет-магазину базується на чітких взаємозв'язках між різними сутностями: клієнтами, товарами та їхніми покупками. MySQL дозволяє побудувати ефективну реляційну архітектуру, яка складається з пов'язаних таблиць: бази користувачів, каталогу книг, реєстру замовлень та деталей конкретних покупок. Використання зовнішніх ключів забезпечує прив'язку кожного замовлення до профілю конкретного покупця, що гарантує цілісність бази даних та безпомилкову ідентифікацію замовника.

Завдяки обраним технологіям, у комплексі ці інструменти дозволяють реалізувати сформовані функціональні та нефункціональні вимоги до системи, забезпечуючи її стабільність, подальшу масштабованість та конкурентоспроможність з можливістю розробки майбутніх покращень сайту [17, с. 192-201].

1.5 Висновки до розділу 1

У першому розділі було здійснено комплексне аналітичне дослідження предметної області – сучасного стану та тенденцій розвитку ринку електронної комерції у секторі книготоргівлі в Україні, а також проведено порівняльний аналіз чинних вебплатформ-аналогів (зокрема, «Yakaboo» та «Книгарня Є»). Проведене дослідження дозволило виявити низку суттєвих недоліків та конструктивних недопрацювань у наявних ринкових рішеннях. Більшість великих маркетплейсів мають перевантажений графічний інтерфейс із надмірною кількістю рекламних

банерів та медіаелементів, що розпорошує увагу покупця, ускладнює навігацію та критично знижує швидкість завантаження сторінок на мобільних пристроях. Окрім цього, базові архітектурні підходи аналогів спираються на застарілу синхронну модель взаємодії, яка вимагає повного перезавантаження вебсторінки при виконанні будь-якої операції (застосування фільтрів, зміна кількості книг, додавання товару до візка), що суттєво погіршує показники користувацького досвіду (UX).

Результатом проведеного аналізу стало обґрунтування та формування специфікації технічних, функціональних і нефункціональних вимог до проектованої комп'ютерної системи інтернет-магазину «CHITANKA». Визначено, що для нівелювання недоліків чинних аналогів розроблювана платформа повинна базуватися на концепції мінімалістичного, інтуїтивно зрозумілого інтерфейсу з фокусом на контент та високу швидкість взаємодії (коефіцієнт Mobile-first).

Сформульовано ключові завдання розробки: впровадження механізмів асинхронного керування даними, реалізація гнучкої системи клієнтської фільтрації без перезавантаження сторінок, створення автономного модуля кошика та забезпечення високого рівня безпеки персональних даних. Такі вимоги стали технічним орієнтиром для наступних етапів проектування та програмної реалізації системи.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Архітектура програмного забезпечення

Для реалізації інтернет-магазину книжкової продукції було обрано класичну трирівневу клієнт-серверну архітектуру [19]. Цей архітектурний підхід є галузевим стандартом для побудови сучасних та надійних систем електронної комерції, оскільки він передбачає чіткий логічний та фізичний поділ програмного продукту на функціонально незалежні компоненти. Структура розробленої

системи базується на виділенні рівня представлення , рівня обробки бізнес-логіки та рівня доступу до даних.

Така структуризація дозволяє ізолювати процеси візуального відображення користувацького інтерфейсу від складних механізмів обробки фінансових транзакцій та збереження конфіденційної інформації. Завдяки чіткому розмежуванню зон відповідальності між рівнями досягається високий ступінь масштабованості системи та спрощується процес її подальшої технічної підтримки чи модернізації програмного коду. Крім того, трирівнева модель суттєво підвищує загальний рівень інформаційної безпеки вебдодатка, оскільки будь-який прямий доступ із клієнтського середовища до бази даних повністю виключається, а вся комунікація проходить через захищений прошарок серверної бізнес-логіки.

2.1.1 Рівні архітектурної моделі вебдодатка

Рівень представлення, що є клієнтською частиною системи, реалізований на базі сучасних стандартів HTML5 та CSS3 із широким застосуванням компонентів фреймворку Bootstrap 5. Головним завданням цього рівня є не лише візуалізація каталогу книг та інтерфейсу взаємодії, а й забезпечення високої динамічності користувацького досвіду без надлишкових запитів до сервера. Особлива роль на даному етапі відведена мові програмування JavaScript, за допомогою якої реалізовано логіку інтерактивного кошика. Використання механізму локального сховища браузера дозволяє зберігати масив обраних товарів безпосередньо на пристрої клієнта, що гарантує миттєве оновлення загальної вартості замовлення та збереження стану покупок при навігації між розділами сайту. Таким чином, рівень представлення виступає автономним середовищем для попереднього формування замовлення та первинної валідації введених користувачем даних [20, с.11-45].

Рівень бізнес-логіки, розроблений мовою PHP, виконує роль центрального координаційного вузла системи та забезпечує безпечну обробку всіх операцій на стороні сервера. Цей рівень відповідає за інтерпретацію запитів, що надходять від клієнта через інтерфейс Fetch API, та управління сесіями користувачів за

допомогою Cookie-файлів. Основна функція серверної частини полягає у проведенні сортування вхідних даних для захисту від потенційних загроз, а також у підготовці та виконанні структурованих запитів до сховища даних. На рівні бізнес-логіки реалізовано алгоритм ідентифікації зареєстрованого користувача та складний процес транзакційного запису замовлення, що передбачає послідовну взаємодію з різними таблицями бази даних для забезпечення логічної цілісності кожної покупки [20, с.114-131].

Рівень доступу до даних, представлений реляційною системою управління базами даних MySQL, забезпечує надійне довгострокове зберігання всієї інформаційної структури інтернет-магазину. Архітектура цього рівня базується на системі взаємопов'язаних таблиць, де окремо фіксуються облікові записи клієнтів у таблиці users, детальні характеристики книжкової продукції у таблиці books, загальні заголовки замовлень у orders та їхні специфічні складові одиниці у order_items. Використання механізмів первинних та зовнішніх ключів гарантує сувору відповідність даних, унеможливлюючи появу ізольованих або некоректних записів. Рівень даних ефективно обробляє запити від сервера на читання та запис інформації, підтримуючи високу швидкість доступу до асортименту та гарантуючи збереження історії взаємодії кожного користувача із торговельним майданчиком [20, с.232-238] .

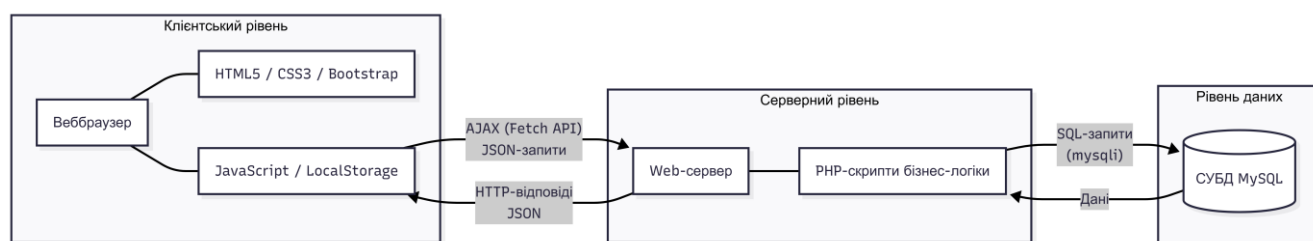


Рисунок 2.1 – Схема клієнт-серверної взаємодії

2.1.2 Механізм взаємодії компонентів та життєвий цикл запити

Динаміка функціонування розробленого вебдодатка ілюструється через життєвий цикл запити під час ключового бізнес-процесу оформлення замовлення.

Цей механізм базується на принципах асинхронної взаємодії, що дозволяє системі опрацьовувати складні транзакції без переривання користувацького досвіду.

Процес розпочинається на рівні представлення, де JavaScript-сценарій оброблює дані з інтерактивної форми та масив об'єктів товарів із локального сховища браузера. Перед ініціалізацією мережевого з'єднання виконується клієнтська валідація, що мінімізує кількість помилкових звернень до сервера. Після успішної перевірки система формує структурований пакет даних у форматі JSON та відправляє асинхронний POST-запит за допомогою інтерфейсу Fetch API до серверного обробника.

На рівні бізнес-логіки серверний скрипт приймає вхідний потік даних та розпочинає процедуру ідентифікації сеансу користувача. Програмний код звертається до масиву куки-файлів для отримання логіна клієнта, після чого здійснює запит до бази даних для визначення відповідного унікального ідентифікатора користувача. Це забезпечує сувору персоналізацію замовлення та дотримання реляційних зв'язків. Далі серверна частина ініціює запис у таблицю замовлень, передаючи санітизовані дані про отримувача та загальну суму. Після успішного створення головного запису PHP-скрипт отримує від СУБД унікальний номер транзакції, який використовується як зовнішній ключ для подальшого циклічного запису кожного окремого товару у таблицю деталей замовлення. Такий двостадійний підхід гарантує цілісність інформації та дозволяє уникнути втрати даних про склад кошика.

Завершальний етап життєвого циклу запиту передбачає формування сервером відповіді у форматі JSON, яка містить статус виконання операції. Клієнтська частина, отримавши підтвердження успішного збереження замовлення, виконує програмне очищення локальної пам'яті браузера від тимчасових даних кошика. Це сигналізує про завершення поточної торгівельної операції, після чого інтерфейс автоматично оновлює стан індикаторів та здійснює перенаправлення користувача на головну сторінку магазину. Використання такої моделі взаємодії дозволяє досягти максимальної ефективності інтерфейсу,

зберігаючи при цьому високу надійність і безпеку обробки даних на стороні сервера [20].

2.2 Проектування бази даних

Основою надійного та стабільного функціонування будь-якої системи електронної комерції є правильно спроектована база даних, яка забезпечує швидкий доступ до інформації, її цілісність та захист від структурних аномалій при додаванні, оновленні чи видаленні записів. Для розроблюваного інтернет-магазину книжкової продукції було обрано реляційну модель даних, реалізовану на базі системи управління базами даних (СУБД) MySQL. Ця модель базується на математичному апараті реляційної алгебри та представляє дані у вигляді взаємопов'язаних двовимірних таблиць (відношень) [21, с.31-49].

Для формалізації структури реляційної бази даних та її підготовки до генерації фізичних SQL-скриптів було використано сучасний індустріальний стандарт опису схем – DBML (Database Markup Language). Ця мова розмітки дозволяє в лаконічній та структурованій формі задокументувати таблиці, типи даних, первинні ключі, а також логічні зв'язки між сутностями системи.

Використання DBML-опису спрощує подальшу підтримку та масштабування бази даних комп'ютерної системи, а також дозволяє автоматично генерувати фізичну схему даних для СУБД MySQL за допомогою спеціалізованих CASE-засобів. Опис розробленої структури бази даних інтернет-магазину наведено в лістингу 2.1.

Лістинг 2.1 – DBML-опис реляційної структури бази даних

```
Table users {  
    id int [pk, increment, note: 'Унікальний ідентифікатор користувача']  
    login varchar(100) [unique, not null, note: 'Логін для автентифікації']
```

```

    password varchar(255) [not null, note: 'Хеш-значення пароля']
}

Table books {
    id int [pk, increment, note: 'Унікальний ідентифікатор
книги']
    title varchar(255) [not null, note: 'Назва книги']
    author varchar(255) [note: 'Автор видання']
    price decimal(10, 2) [not null, note: 'Вартість одиниці
товару']
    category varchar(100) [note: 'Категорія літератури для
фільтрації']
    image_url varchar(255) [note: 'Шлях до зображення
обкладинки']
}

Table orders {
    id int [pk, increment, note: 'Унікальний ідентифікатор
замовлення']
    user_id int [not null, note: 'Зовнішній ключ, зв'язок із
таблицею users']
    total_price decimal(10, 2) [not null, note: 'Загальна сума
замовлення']
    customer_name varchar(100) [not null, note: 'Ім'я та прізвище
отримувача']
    phone varchar(20) [not null, note: 'Контактний номер
телефону']
    address text [not null, note: 'Адреса доставки']
    created_at datetime [default: `now()`, note: 'Дата та час
створення запису']
}

Table order_items {
    id int [pk, increment, note: 'Ідентифікатор рядка
специфікації замовлення']
    order_id int [not null, note: 'Зовнішній ключ, зв'язок із
таблицею orders']
    book_id int [not null, note: 'Зовнішній ключ, зв'язок із
таблицею books']
    quantity int [not null, default: 1, note: 'Кількість
примірників книги']
    price decimal(10, 2) [not null, note: 'Фіксована ціна на
момент купівлі']
}

Ref: orders.user_id > users.id [delete: cascade]
Ref: order_items.order_id > orders.id [delete: cascade]
Ref: order_items.book_id > books.id

```

У результаті проєктування було створено оптимізовану архітектуру бази даних, що складається з чотирьох основних взаємопов'язаних таблиць:

– «users» – призначена для зберігання облікових даних клієнтів інтернет-магазину, що пройшли процедуру реєстрації. Структура таблиці включає унікальний ідентифікатор `id` цілочисельного типу, логін для авторизації в системі (`login`), зашифрований пароль доступу (`password`) ;

– «books» – виконує роль електронної вітрини магазину та зберігає вичерпну інформацію про асортимент. Основним ідентифікатором запису є первинний ключ `id`, назва видання `title` (назва книги) та `author` (автор), рік випуску (`publish_year`), для фіксації вартості товару поле `price` типу `DECIMAL`, що гарантує фінансову точність при грошових розрахунках, література записана через поле `category`, візуальне представлення товару забезпечується полем `image`, яке містить шлях до файлу обкладинки книги на сервері;

– «orders» – накопичує загальну інформацію про оформлені покупки та контактні дані клієнта на момент транзакції. Первинним ключем виступає поле `id`. Для забезпечення зв'язку з профілем покупця використовується поле `user_id`. Зберігаються персональні та логістичні дані отримувача: `name`, `email`, `phone` (контактний номер), `city` (місто доставки) та `address` (точна адреса). Загальна сума замовлення фіксується у полі `total_price`, системне поле `created_at` призначене для точної автоматичної фіксації дати та часу створення замовлення, а також поле `payment_method` для збереження обраного способу оплати;

– «order_items» – містить деталізовану інформацію про склад кожного конкретного чека. унікальний номер системного запису (`id`), номер прив'язаного замовлення (`order_id`), ідентифікатор купленої книги (`book_id`), кількість замовлених примірників цього товару (`quantity`), ціна однієї одиниці товару на момент підтвердження покупки.

Розробка реляційної моделі даних передбачає не лише створення ізольованих таблиць, але й встановлення строгих логічних зв'язків між ними. Це є важливою вимогою для забезпечення цілісності зв'язків бази даних, уникнення дублювання інформації та захисту від появи некоректних записів.

На рисунку 2.2 наведено ER-діаграму спроектованої бази даних. Вона наочно демонструє архітектуру системи, яка складається з чотирьох основних

сутностей. Зв'язки між ними фізично реалізовані за допомогою механізму зовнішніх ключів і відображені на схемі за нотацією «вороняча лапка», що графічно вказує на тип відношення «один до багатьох».

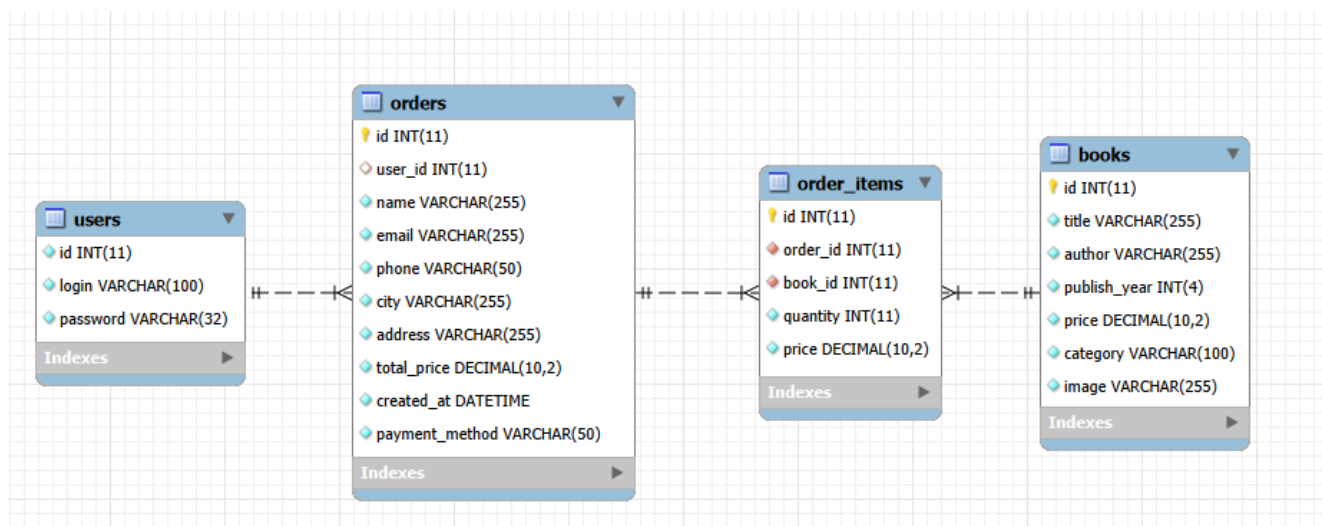


Рисунок 2.2 – ER-діаграма реляційної бази даних інтернет-магазину

Оскільки бізнес-логіка електронної комерції передбачає, що одне замовлення може містити багато різних книг, а одна й та сама книга може фігурувати у безлічі різних замовлень, між сутностями `orders` та `books` виникає відношення «багато до багатьох». Для його реалізації у реляційній базі даних розроблено проміжну таблицю `order_items`. Таким чином, складна архітектура була розділена на три базові зв'язки типу «один до багатьох», які налаштовані на рівні СУБД, а саме:

– зв'язок між користувачами та замовленнями - кожен зареєстрований клієнт може мати безліч оформлених покупок, проте кожне конкретне замовлення завжди належить лише одному користувачу. Для забезпечення цього відношення у таблиці `orders` створено зовнішній ключ із назвою `fk_user` (рис. 2.8). Це обмеження пов'язує поле `user_id` таблиці замовлень із первинним ключем `id` батьківської таблиці `users`.

 Table Name: orders	
Foreign Key Name	Referenced Table
fk_user	`vstupvadam`.`users`

Рисунок 2.3 – Конфігурація зовнішнього ключа `fk_user` у таблиці `orders`

– зв'язок між замовленням та його деталями – для того, щоб система розуміла, які саме товари належать до конкретного чека, налаштовано зв'язок між загальною таблицею замовлень та її деталізацією. У проміжній таблиці `order_items` створено зовнішній ключ `fk_order` (рис. 2.4). Він посилається на таблицю `orders`, гарантуючи, що поле `order_id` у кожному рядку купленого товару чітко відповідає існуючому ідентифікатору з таблиці загальних замовлень.

– зв'язок між каталогом книг та деталями замовлення Аналогічно, для прив'язки конкретного фізичного товару до рядка замовлення, в таблиці `order_items` створено другий зовнішній ключ – `fk_book` (рис. 2.4). Він посилається на таблицю `books`, пов'язуючи поле `book_id` із первинним ключем `id` каталогу товарів.

Foreign Key Name	Referenced Table
fk_order	`vstupvadam`.`orders`
fk_book	`vstupvadam`.`books`

Рисунок 2.4 – Конфігурація зовнішніх ключів у транзитній таблиці `order_items`

Налаштування цих зовнішніх ключів є обов'язковим для стабільності бекенд-частини інтернет-магазину. Завдяки їм, СУБД автоматично контролює цілісність даних, таким чином система фізично не дозволить програмному коду створити помилковий запит, наприклад: замовлення з неіснуючою книгою або товару з каталогу.

2.3 Проектування інтерфейсу користувача

Головною метою цього етапу є створення інтуїтивно зрозумілого і зручного інтерфейсу користувача (UI) і користувацького досвіду (UX), яка дозволить клієнту швидко знаходити потрібну книгу та без перешкод оформлювати замовлення [22]. Архітектура інтерфейсу базується на гнучкій сітці, що гарантує коректне відображення всіх елементів як на десктопних моніторах, так і на екранах мобільних пристроїв. Для візуалізації структури сайту на етапі проектування було створено схематичні макети ключових сторінок платформи.

Загальна структура веб додатка поділяється на чотири основні функціональні зони: головна сторінка, каталог товарів, сторінка окремого товару та сторінка оформлення замовлення. Основним елементом для всіх сторінок є верхня навігаційна панель, яка містить логотип, посилання на основні розділи та кнопку інтерактивного кошика з динамічним бейджем кількості товарів.

Головна сторінка виконує функцію вітрини магазину та формує перше враження користувача. Її структуру спроектовано за класичною моделлю комерційної посадкової сторінки, що складається з послідовних інформаційних блоків:

- слайдер: верхній графічний блок для демонстрації актуальних акцій або новинок;
- наші переваги: інформаційний блок, який містить короткі тези та опис магазину;
- рекомендовані товари: блок із картками найпопулярніших книг, що дозволяє користувачу здійснити швидку покупку одразу з головної сторінки;
- про нас: коротка текстова інформація про книгарню;
- соціальна інформація: нижній блок із контактними даними, посиланнями на соціальні мережі та правовою інформацією.

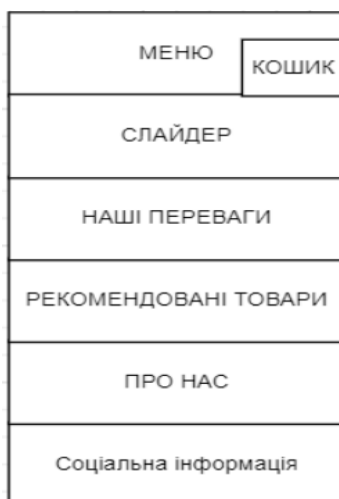


Рисунок 2.5 – Дизайн сторінки «Головна»

Сторінка каталогу є основним робочим простором користувача для пошуку літератури. Інтерфейс цього розділу спроектовано з акцентом на зручність обробки та фільтрації великого масиву даних:

- блок "Фільтр": розміщений у лівій частині екрана надає користувачеві інструменти для сортування книг за жанрами;
- блок "Книги": центральна область, де товари виводяться у вигляді структурованої адаптивної сітки, кожна картка товару містить обкладинку книги, назву, автора, ціну та кнопку швидкого додавання до кошика, що дозволяє робити покупки без переходу на сторінку конкретного товару.



Рисунок 2.6 – Дизайн сторінки «Каталог товарів»

Сторінка товару призначена для детального ознайомлення з конкретним виданням перед прийняттям рішення про покупку. Структура макета розділена на візуальну та інформаційну складові:

- блок товару: містить збільшене зображення обкладинки, детальний опис, характеристики книги (рік видання, автор, видавництво);
- блок оплати: спроектований як акцентна зона праворуч яка містить актуальну ціну та цільову кнопку додавання товару до кошика.

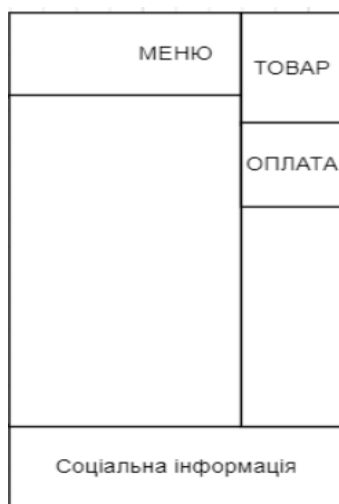


Рисунок 2.7 – Дизайн бокової панелі «Товарів»

На сторінці оформлення інтерфейс чітко розділений на дві функціональні колонки:

– заповнення даних: форма для введення персональних та логістичних даних користувача, форма має налаштовану клієнтську валідацію для підсвічування некоректно заповнених полів.

– оплата та підсумок: динамічний блок, що генерується за допомогою JavaScript на основі даних з localStorage, він відображає перелік обраних книг, їх кількість, загальну суму до сплати та кнопки для вибору способу оплати. Завершується блок великою кнопкою «Оплатити», яка ініціює асинхронну відправку зібраних даних на сервер



Рисунок 2.8 – Дизайн сторінки «Оплата товарів»

Запропонована структура інтерфейсу відповідає сучасним стандартам електронної комерції. Вона забезпечує логічний маршрут користувача від моменту входу на сайт до успішного підтвердження транзакції, роблячи процес купівлі книг максимально інтуїтивним та швидким.

2.4 Моделювання поведінки системи

Аналіз динамічних аспектів функціонування інтернет-магазину включає у себе моделювання поведінки системи, яка дозволяє детально описати логіку взаємодії користувача з веб ресурсом, визначити послідовність внутрішніх процесів та встановити протоколи комунікації між клієнтською і серверною частинами додатка. Для розв'язання цих завдань у межах даної роботи було використано мову графічного моделювання UML, яка є стандартом у проектуванні складних програмних систем [23].

Першим етапом моделювання є визначення ролей користувачів та їхніх функціональних можливостей. Для цього було розроблено діаграму прецедентів, яка візуалізує межі системи та описує ключові сценарії взаємодії для різних типів

акторів – неавторизованого гостя та зареєстрованого клієнта. Дана модель дозволяє чітко розмежувати доступ до функціонала перегляду товарів та критичних операцій оформлення замовлення.

Для моделювання поведінки розроблюваної системи електронної комерції було обрано діаграму прецедентів. Ця діаграма дозволяє визначити основних дійових осіб (акторів) та перелік функцій, які вони можуть виконувати під час взаємодії з веб додатком.

У проєкті виділено дві основні ролі користувачів:

- гість – неавторизований відвідувач сайту. Його функціональні можливості обмежені ознайомленням з асортиментом та підготовкою до покупки. Гість може здійснювати перегляд каталогу , використовувати систему динамічної фільтрації за категоріями , додавати обрані книги до інтерактивного кошика та проходити процедуру реєстрації або входу до системи;
- авторизований користувач – ідентифікований системою відвідувач, який має доступ до ініціювання процесу оформлення замовлення.

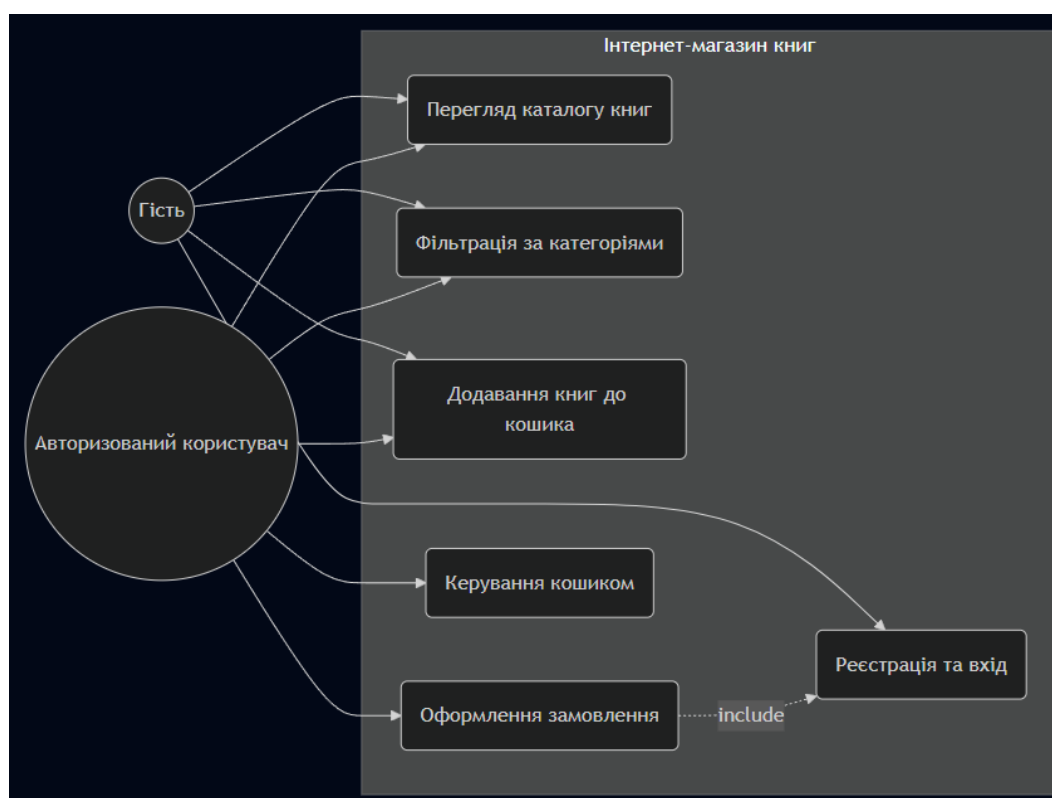


Рисунок 2.9 – Діаграма прецедентів системи

Для більш глибокого розуміння внутрішньої логіки найважливіших бізнес-процесів, зокрема алгоритму купівлі книжкової продукції, було застосовано діаграму діяльності. Вона демонструє покроковий маршрут користувача від моменту наповнення кошика до отримання підтвердження про успішну транзакцію. Ця діаграма дозволяє врахувати всі логічні розгалуження, такі як перевірка стану авторизації та результати валідації введених даних, що гарантує стабільність роботи системи при виникненні помилкових сценаріїв.

Для детального аналізу логіки роботи ключового бізнес-процесу системи – оформлення замовлення – було розроблено діаграму діяльності. Ця діаграма відображає динаміку взаємодії користувача з веб додатком, описуючи послідовність дій та умови прийняття рішень системою.

Алгоритм процесу оформлення замовлення включає наступні етапи:

- аналіз стану кошика: процес розпочинається з перевірки вмісту локального сховища, якщо кошик порожній, система блокує можливість переходу до оплати та виводить відповідне повідомлення, у разі наявності товарів користувач отримує доступ до кнопки оформлення;

- перевірка прав доступу: точкою розгалуження є стан авторизації, якщо клієнт не увійшов до облікового запису, система перенаправляє його на сторінку входу, тільки після успішної ідентифікації стає доступною сторінка `checkout.php`;

- валідація та відправка: на етапі заповнення форми JavaScript здійснює первинну перевірку полів на стороні клієнта, при виявленні помилок система підсвічує відповідні поля, після успішної валідації дані асинхронно відправляються до серверного обробника;

- транзакційна обробка: на серверному рівні PHP-скрипт координує послідовний запис даних у дві пов'язані таблиці бази даних;

- фіналізація: після отримання підтвердження від бази даних система програмно очищує локальний кошик користувача, виводить діалогове вікно з подякою та здійснює автоматичний редирект на головну сторінку магазину.

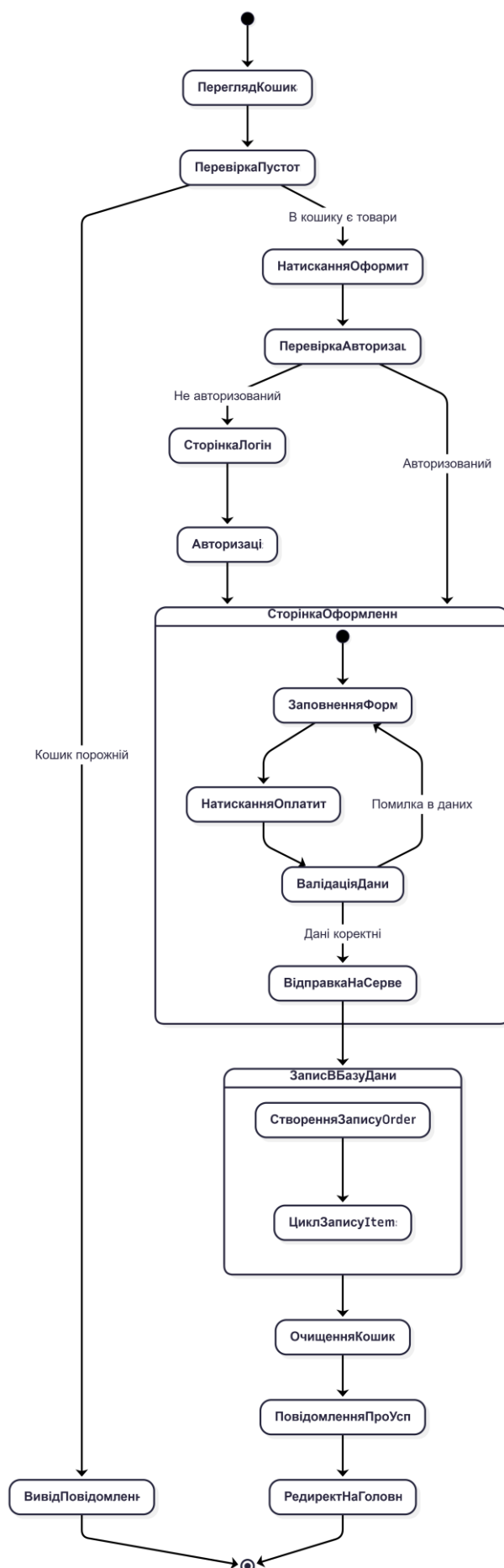


Рисунок 2.10 – Діаграма послідовностей

Діаграма послідовностей фокусується на обміні повідомленнями між браузером користувача, серверними скриптами на мові PHP та реляційною базою даних MySQL. Модель наочно демонструє механізм асинхронної відправки даних через Fetch API та транзакційний процес збереження інформації, що підтверджує цілісність та надійність реалізованої клієнт-серверної архітектури.

Для деталізації технічної взаємодії між об'єктами, було розроблено діаграму послідовностей (Sequence Diagram). Ця модель візуалізує обмін повідомленнями між чотирма основними учасниками процесу: користувачем, клієнтським сценарієм (JavaScript), серверним обробником (PHP) та системою управління базами даних (MySQL) [24].

Взаємодія в системі організована за наступним алгоритмом:

- ініціація та клієнтська перевірка: після натискання користувачем цільової кнопки, JavaScript-сценарій перехоплює подію та виконує первинну валідацію введених даних, у разі успіху формується структурований пакет даних у форматі JSON, який включає персональну інформацію та масив об'єктів із локального сховища кошика;

- асинхронна передача даних: використовуючи інтерфейс Fetch API, клієнтська частина відправляє асинхронний POST-запит до серверного файлу `save_order.php`, це дозволяє уникнути перезавантаження сторінки та забезпечити безперервність користувацького досвіду;

- ідентифікація та серверна логіка: серверний обробник ідентифікує поточну сесію користувача через Cookie-файли та звертається до таблиці `users` для отримання унікального ідентифікатора клієнта (`user_id`), що забезпечує цілісність зв'язків у базі даних;

- транзакційний запис у базу даних: процес збереження замовлення розділений на два етапи, спочатку створюється запис у таблиці `orders`, після чого СУБД повертає ідентифікатор щойно створеного рядка (`insert_id`), на другому етапі PHP-скрипт у циклічному режимі записує кожну позицію з кошика до таблиці `order_items`, прив'язуючи її до отриманого номера замовлення;

– завершення транзакції: після отримання підтвердження від бази даних сервер відправляє клієнту статус успішного завершення операції. JavaScript очищує локальну пам'ять браузера (localStorage) та інформує користувача про успішне прийняття замовлення.

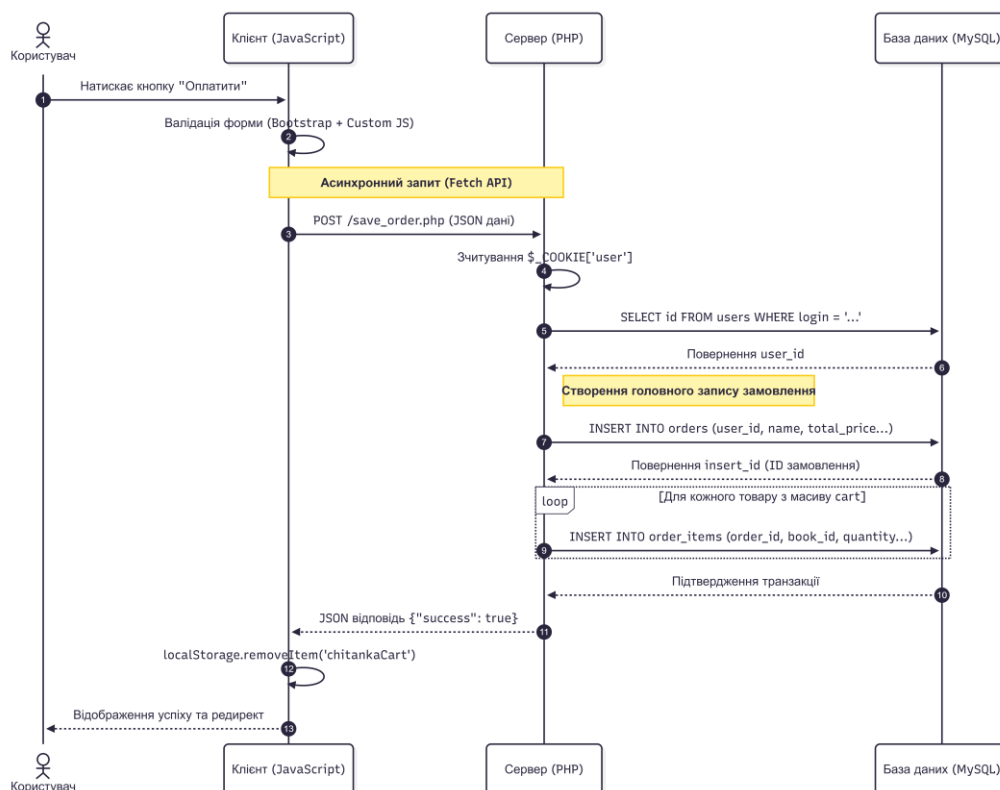


Рисунок 2.11 – Діаграма послідовностей взаємодії компонентів

Діаграма (рис.2.11) підтверджує ефективність обраної трирівневої архітектури, де кожен рівень виконує чітко визначену роль, мінімізуючи навантаження на мережеві ресурси та гарантуючи надійність збереження інформації.

2.5 Обґрунтування апаратних вимог до комп'ютерної системи

Оскільки розроблювана комп'ютерна система інтернет-магазину «CHITANKA» побудована за трирівневою клієнт-серверною архітектурою, її

стабільне функціонування залежить від апаратних характеристик двох основних контурів: серверного обладнання (де виконуються PHP-скрипти та працює СУБД MySQL) та клієнтських пристроїв (комп'ютерів, ноутбуків або смартфонів користувачів).

З огляду на архітектурне рішення перенести частину обчислень на сторону клієнта (обробка кошика через LocalStorage та асинхронна фільтрація за допомогою JavaScript), навантаження на сервер є оптимізованим. Це дозволяє системі стабільно працювати навіть на базі недорогого хостингу або віртуального сервера.

Сервер повинен забезпечувати одночасну обробку HTTP-запитів від багатьох користувачів, роботу інтерпретатора PHP 8.x та швидкий доступ до дискової підсистеми для виконання SQL-запитів.

Мінімальні апаратні вимоги до сервера:

- процесор: 1 ядро з тактовою частотою від 2.0 ГГц;
- оперативна пам'ять: 1 Гб;
- дисковий простір: 10 Гб вільного місця;
- мережевий інтерфейс: пропускна здатність від 10 Мбіт/с.

Рекомендовані апаратні вимоги до сервера (для стабільної роботи при високому навантаженні):

- процесор: 2 або 4 ядра з тактовою частотою від 2.4 ГГц;
- оперативна пам'ять: 4 Гб або більше;
- дисковий простір: 20–40 Гб на швидкому твердотільному накопичувачі типу SSD;
- мережевий інтерфейс: пропускна здатність від 100 Мбіт/с.

Оскільки інтерфейс системи є адаптивним і побудований на базі фреймворку Bootstrap 5, жорстких вимог до заліза користувача немає. Головна умова – здатність пристрою запускати сучасні веббраузери, які підтримують стандарт ES6+ для коректного виконання JavaScript-скриптів кошика та фільтрації.

Мінімальні вимоги до пристрою користувача (ПК, ноутбук, смартфон):

- процесор: архітектура x86/x64 або ARM із частотою від 1.2 ГГц;
- оперативна пам'ять: 2 Гб;
- дисплей: роздільна здатність від 320×480 пікселів;
- підключення до мережі Інтернет: швидкість від 1 Мбіт/с.

Рекомендовані вимоги до пристрою користувача:

- процесор: багатоядерний процесор із частотою від 2.0 ГГц;
- оперативна пам'ять: 4 Гб або більше;
- дисплей: роздільна здатність 1920×1080 або від 1080×2400;
- підключення до мережі Інтернет: 3G/4G/5G або широкосмуговий доступ від 10 Мбіт/с.

Таким чином, розроблена комп'ютерна система є невибагливою до апаратних ресурсів, що робить її розгортання та експлуатацію економічно вигідною для кінцевого замовника.

2.6 Висновок до розділу 2

Другий розділ присвячено етапу системного проєктування, архітектурного моделювання та розробки інформаційного забезпечення інтернет-магазину. На основі сформованих вимог було обґрунтовано доцільність використання класичної трирівневої клієнт-серверної архітектури вебдодатка, яка передбачає суворе розділення рівня представлення даних, рівня бізнес-логіки та рівня збереження даних. Таке рішення гарантує високу масштабованість системи, спрощує її подальшу модернізацію та підвищує загальну відмовостійкість і безпеку інформаційних потоків між компонентами.

Для формалізації та алгоритмізації логіки функціонування платформи було побудовано комплекс графічних моделей мовою UML. За допомогою діаграми прецедентів чітко розмежовано ролі користувачів та їхні права доступу до модулів системи. Діаграми діяльності та послідовностей дозволили детально

описати покрокові сценарії асинхронного обміну даними між фронтом та бекендом під час автентифікації користувачів, динамічного оновлення інтерфейсу та транзакційного оформлення замовлень. Також у розділі було спроектовано логічну схему реляційної бази даних у середовищі MySQL. Усі розроблені таблиці було приведено до третьої нормальної форми, оптимізовано за типами даних та пов'язано між собою через зовнішні ключі з налаштуванням правил каскадного видалення, що повністю виключає ризик появи ізольованих записів та забезпечує цілісність даних на рівні СУБД. Додатково структуру сховища зафіксовано за допомогою індустріального стандарту DBML-опису.

Також розглянуто обґрунтування апаратних вимог до технічних засобів комп'ютерної системи. Завдяки архітектурному рішенню перенести частину обчислювального навантаження на сторону клієнта, було досягнуто оптимізації використання потужностей сервера. Сформульовані технічні вимоги до серверного обладнання та пристроїв користувачів доводять, що розроблена комп'ютерна система є невибагливою до ресурсів заліза, може стабільно функціонувати на базі стандартного недорогого хостингу і є економічно ефективною для впровадження.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Розробка клієнтської частини та каталогу

Розробка клієнтської частини та каталогу передбачає реалізацію адаптивної верстки товарних карток та динамічної фільтрації книг за категоріями. Каталог є основним робочим простором інтернет-магазину, тому його архітектура повинна витримувати високі навантаження та забезпечувати комфортну взаємодію з великим обсягом даних.

Для забезпечення адаптивності інтерфейсу було використано сіткову систему (Grid System) фреймворку Bootstrap 5 [25]. Каталог побудовано за

принципом контейнерної верстки, де кожна книга представлена у вигляді окремого UI-компонента – картки (Card). Завдяки застосуванню адаптивних класів сітки (наприклад, col-12 для мобільних пристроїв, col-md-6 для планшетів та col-lg-3 для десктопних моніторів), кількість товарів, що відображаються в одному горизонтальному ряду, автоматично змінюється залежно від фізичної роздільної здатності екрана пристрою клієнта. Це повною мірою реалізує стратегію Mobile-first, гарантуючи відсутність горизонтального скролінгу та деформації контенту.

Виведення списку книг здійснюється шляхом звернення до бази даних та циклічної обробки отриманого результату. Замість статичного прописування кожного товару, використовується цикл while, який генерує HTML-структуру для кожного рядка з таблиці books. Код, що відповідає за генерацію карток товарів, наведено у лістингу 3.1.

Лістинг 3.1 – Серверна генерація каталогу товарів

```
<?php while ($book = $result->fetch_assoc()): ?>
  <div class="col-lg-4 col-sm-6 mb-3 product" data-category="<?=
htmlspecialchars($book['category']) ?>">
    <div class="product-card">
      <div class="product-thumb">
        <a href="product.php?id=<?= $book['id'] ?>">
![<?= htmlspecialchars($book['title']) ?>](assets/img/books/<?= htmlspecialchars($book['image']) ?>)

```

```

        data-price="<?= $book['price'] ?>">
        <i class="fas fa-shopping-cart"></i>
    </button>
</div>
</div>
</div>
</div>
<?php endwhile; ?>

```

У лістингу 3.1 продемонстровано використання циклу `while`, який ітерує отриманий з бази даних масив записів `$result`. Для забезпечення крос-сайтової безпеки усі текстові дані, що виводяться на екран (назва, ціна, посилання на зображення), обробляються вбудованою функцією `htmlspecialchars()`.

Адаптивність розмітки реалізовано за допомогою класів фреймворку Bootstrap (`col-lg-4 col-sm-6`), що гарантує коректне відображення карток як на мобільних пристроях, так і на десктопних моніторах. Важливим архітектурним рішенням є використання `data`-атрибутів (зокрема `data-category`, `data-id`, `data-title`, `data-price`). Вони дозволяють зберігати метадані товару безпосередньо в DOM-дереві, що робить їх доступними для подальшої обробки клієнтськими JavaScript-сценаріями без додаткових звернень до сервера [26].

Для підвищення ергономічності інтерфейсу була розроблена система динамічної фільтрації каталогу. Вона дозволяє користувачеві сортувати книги за категоріями у режимі реального часу, мінаючи етап перезавантаження сторінки. Логіку цього процесу наведено у лістингу 3.2.

Лістинг 3.2 – Сценарій динамічної фільтрації карток товарів

```

document.addEventListener("DOMContentLoaded", function () {
    const checkboxes = document.querySelectorAll(".filter-block
input[type='checkbox']");
    const products = document.querySelectorAll(".product-
card");
    function filterProducts() {
        const selectedCategories = Array.from(checkboxes)
            .filter(checkbox => checkbox.checked)
            .map(checkbox => checkbox.id);

        products.forEach(product => {
            const productCategory = product.closest(".col-lg-
4").dataset.category;

```

```

        if (selectedCategories.length === 0 ||
selectedCategories.includes(productCategory)) {
            product.closest(".col-lg-4").style.display =
"";
        } else {
            product.closest(".col-lg-4").style.display =
"none";
        }
    });
}
checkboxes.forEach(checkbox => {
    checkbox.addEventListener("change", filterProducts);
});
filterProducts();
});

```

Лістинг, наведений у лістингу 3.2, ініціалізується після повного завантаження об'єктної моделі документа (DOMContentLoaded). Функція filterProducts формує масив selectedCategories, до якого потрапляють ідентифікатори лише тих категорій (чекбоксів), які були обрані користувачем. Далі скрипт виконує перебір усіх карток товарів на сторінці. За допомогою методу closest(".col-lg-4") функція знаходить батьківський контейнер поточної картки та зчитує значення його атрибута dataset.category. Якщо масив обраних категорій порожній (фільтри не застосовано) або категорія товару присутня у масиві, товар залишається видимим. У протилежному випадку властивості style.display присвоюється значення "none", що програмно приховує елемент.

Обробка покупок на стороні клієнта вимагає надійного механізму оновлення інтерфейсу відповідно до змін у масиві обраних товарів. За це відповідає функція рендерингу кошика, наведена у лістингу 3.3.

Лістинг 3.3 – Функція рендерингу та перерахунку вмісту кошика

```

function renderCart() {
    if (!cartTable) return;
    cartTable.innerHTML = "";
    let totalPrice = 0;
    cart.forEach(product => {
        const productTotal = product.price * product.quantity;
        totalPrice += productTotal;
        const row = `
            <tr>
                <td>${product.name}</td>

```

```

        <td>${product.price}</td>
        <td>&times;${product.quantity}</td>
        <td>
            <button class="btn btn-danger remove-from-
cart" data-id="${product.id}">
                <i class="fa-solid fa-xmark"></i>
            </button>
        </td>
    </tr>
    `;
    cartTable.innerHTML += row;
});
if (totalPriceElement){
    totalPriceElement.textContent = `${totalPrice}`;
}

document.querySelectorAll(".remove-from-
cart").forEach(button => {
    button.addEventListener("click", () => {
        const productId = button.getAttribute("data-id");
        removeFromCart(productId);
    });
});
updateBadge();
}

```

Алгоритм функції `renderCart` розпочинається з очищення попереднього вмісту таблиці кошика (`cartTable.innerHTML = ""`). Далі, за допомогою методу `forEach`, скрипт проходить по глобальному масиву `cart`. Для кожної товарної позиції обчислюється проміжна вартість (`productTotal`), яка додається до загальної суми замовлення (`totalPrice`). Паралельно формується HTML-розмітка нового рядка таблиці, що містить назву книги, ціну, кількість примірників та кнопку видалення товару з унікальним атрибутом `data-id`. Після завершення циклу загальна сума виводиться у відповідний елемент інтерфейсу. Завершальним кроком є призначення слухачів подій (Event Listeners) на щойно згенеровані кнопки видалення та виклик функції `updateBadge()`, яка оновлює числовий індикатор кількості товарів у навігаційній панелі.

Окрім основного функціоналу каталогу, клієнтська частина містить низку допоміжних скриптів для оптимізації взаємодії з користувачем. Програмний код цих мікроінтеракцій наведено у лістингу 3.4.

Лістинг 3.4 – Сценарії обробки навігації та валідації форм

```

window.addEventListener('scroll', function() {
    document.getElementById('header-
nav').classList.toggle('headernav-scroll', window.scrollY > 180);
});
(function () {
    'use strict';
    const forms = document.querySelectorAll('.needs-
validation');
    Array.prototype.slice.call(forms).forEach(function (form) {
        form.addEventListener('submit', function (event) {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();
            }
            form.classList.add('was-validated');
        }, false);
    });
})();

```

У першій частині лістингу 3.4 реалізовано відстеження події прокручування вікна браузера (scroll). Якщо вертикальний відступ перевищує 180 пікселів, до навігаційної панелі автоматично додається CSS-клас `headernav-scroll`, що дозволяє змінити її візуальне оформлення (наприклад, закріпити у верхній частині екрана або змінити колір фону) для забезпечення постійного доступу до меню.

Друга частина коду використовує механізм самовикликуваної анонімної функції (IIFE) для налаштування клієнтської валідації вебформ. Скрипт перехоплює подію `submit` для всіх форм із класом `.needs-validation`. Якщо метод `checkValidity()` виявляє порушення правил заповнення полів, відправка форми блокується методами `preventDefault()` та `stopPropagation()`, а до форми додається клас `was-validated`, який активує вбудовані у Bootstrap механізми візуальної індикації помилок введення даних.

3.2.1 Реалізація локального збереження даних за допомогою Web Storage API

У сучасних системах електронної комерції забезпечення стійкості даних користувацького вибору є критично важливою вимогою. Користувач повинен мати можливість повернутися до перерваного процесу покупок навіть після перезавантаження сторінки або закриття браузера. Для вирішення цього завдання

у межах розроблюваного вебдодатка було обрано механізм `localStorage`, який є частиною стандарту `Web Storage API`.

На відміну від традиційного використання `Cookie`-файлів, робота з `localStorage` дозволяє зберігати дані обсягом до 5 МБ безпосередньо в пісочниці браузера. Це виключає надлишкову передачу інформації про вміст кошика в заголовках кожного `HTTP`-запиту, що позитивно впливає на швидкість завантаження сторінок та знижує навантаження на сервер.

Програмна реалізація механізму синхронізації внутрішнього стану кошика із локальним сховищем браузера наведена у лістингу 3.5.

Лістинг 3.5 – Код ініціалізації та синхронізації кошика з `LocalStorage`

```
let cart = JSON.parse(localStorage.getItem('chitankaCart')) ||
[];
function saveCartToStorage() {
    localStorage.setItem('chitankaCart', JSON.stringify(cart));
}
function syncCartState() {
    saveCartToStorage();
    renderCart();
    updateBadge();}
```

На початку лістингу 3.5 продемонстровано процес ініціалізації змінної `cart`. Оскільки в `localStorage` дані зберігаються виключно у вигляді рядків, для відновлення структури масиву об'єктів використовується метод `JSON.parse()`. Застосування логічного оператора «`||`» дозволяє ініціалізувати порожній масив у разі першого візиту користувача, коли запис із ключем `chitankaCart` ще відсутній у пам'яті браузера.

Функція `saveCartToStorage()` відповідає за серіалізацію поточного стану масиву об'єктів (назви книг, ціни, `ID` та кількість) у формат `JSON`-рядка за допомогою методу `JSON.stringify()`. Це перетворення є необхідним, оскільки сховище `Web Storage` не підтримує нативне збереження об'єктів `JavaScript`.

Функція `syncCartState()` виступає в ролі координатора: вона гарантує, що будь-яка маніпуляція з даними (додавання нової книги або видалення позиції) буде одночасно зафіксована у локальному сховищі та візуалізована в інтерфейсі

через виклики `renderCart()` та `updateBadge()`. Така реалізація забезпечує цілісність даних та дозволяє реалізувати вимогу щодо синхронізації кошика між різними вкладками браузера, оскільки подія зміни `localStorage` може відстежуватися в межах одного домену.

Використання `Web Storage API` дозволило створити незалежну від сервера систему зберігання клієнтських даних, що суттєво покращує відгук інтерфейсу та забезпечує безперервність користувацького досвіду при роботі з каталогом.

3.2.2 Логіка управління станом кошика та функції додавання і видалення товарів

Для забезпечення коректної роботи інтерактивного кошика було розроблено набір функцій, які керують станом масиву товарів у реальному часі. Основна складність реалізації полягає у необхідності перевірки наявності дублікатів: система повинна ідентифікувати, чи додається новий примірник уже обраної книги, чи створюється новий запис у списку покупок.

Програмна реалізація функцій додавання та видалення товарів наведена у лістингу 3.6.

Лістинг 3.6 – Логіка управління масивом товарів у кошику

```
function addToCart(productId, productName, productPrice) {
    const existingProduct = cart.find(item => item.id ===
productId);
    if (existingProduct) {
        existingProduct.quantity += 1;
    } else {
        cart.push({
            id: productId,
            name: productName,
            price: productPrice,
            quantity: 1,
        });
    }
    renderCart();
}

function removeFromCart(productId) {
    cart = cart.filter(item => item.id !== productId);
    renderCart();
}

document.querySelectorAll(".add-to-cart").forEach(button => {
    button.addEventListener("click", event => {
```

```

        event.preventDefault();
        const productId = button.getAttribute("data-id");
        const productCard = button.closest(".product-card");
        if (!productCard) return;
        const productName =
productCard.querySelector(".product-details a").textContent;
        const productPrice =
parseFloat(productCard.querySelector(".product-
price").textContent.replace("$", ""));
        addToCart(productId, productName, productPrice);
    });
});

```

У лістингу 3.6 реалізовано алгоритм функції `addToCart`, яка приймає унікальний ідентифікатор, назву та ціну книги. На першому етапі функція використовує метод масиву `find()` для пошуку об'єкта, властивість `id` якого збігається із переданим значенням `productId`. У разі успішного знаходження (умова `if (existingProduct)`) система інкрементує властивість `quantity`, що дозволяє уникнути дублювання однакових позицій у таблиці кошика. Якщо товар додається вперше, метод `push()` створює новий об'єкт із початковою кількістю 1.

Функція `removeFromCart` реалізує зворотну дію – виключення товару з масиву. Для цього застосовується метод `filter()`, який створює новий масив, що містить усі елементи, крім того, чий ідентифікатор збігається із цільовим. Такий підхід гарантує цілісність даних та дозволяє миттєво видалити всі примірники обраної позиції.

Окрему увагу приділено обробнику подій для кнопок додавання. Скрипт виконує ітерацію за допомогою `forEach` по всіх елементах із класом `.add-to-cart`. При натисканні на кнопку метод `closest(".product-card")` дозволяє піднятися вгору по DOM-дереву до батьківського контейнера конкретної книги. Це необхідно для точного зчитування текстового вмісту назви та ціни. Отримана ціна обробляється функцією `parseFloat()` із попереднім видаленням символу валюти через метод `replace()`, що перетворює рядок на числове значення, придатне для математичних розрахунків.

Після кожної операції з масивом обов'язково викликається функція `renderCart()`, яка ініціює оновлення візуального стану інтерфейсу, забезпечуючи реактивність системи.

3.2.3 Динамічне оновлення інтерфейсу в реальному часі

Одною з основних вимог до сучасних інтерфейсів електронної комерції є забезпечення миттєвого візуального відгуку на дії користувача. У розробленому вебдодатку реалізовано механізм динамічного оновлення компонентів кошика, що дозволяє відображати зміни у кількості товарів та загальній вартості замовлення без перезавантаження сторінки. Це досягається шляхом прямої маніпуляції з об'єктною моделлю документа (DOM) за допомогою JavaScript.

Програмна реалізація функцій рендерингу та динамічного підрахунку показників наведена у лістингу 3.7.

Лістинг 3.7 – Функції візуалізації стану кошика та оновлення індикаторів

```
function updateBadge() {
    if(!cartBadge) return;
    let totalItems = 0;
    cart.forEach(product => {
        totalItems += product.quantity;
    });
    cartBadge.textContent = totalItems;
    cartBadge.style.display = totalItems > 0 ? "inline-block" :
"none";
}

function renderCart() {
    if (!cartTable) return;
    cartTable.innerHTML = "";
    let totalPrice = 0;

    cart.forEach(product => {
        const productTotal = product.price * product.quantity;
        totalPrice += productTotal;

        const row = `
            <tr>
                <td>${product.name}</td>
                <td>${product.price}</td>
                <td>&times; ${product.quantity}</td>
                <td>
                    <button class="btn btn-danger remove-from-
cart" data-id="${product.id}">
```

```

                <i class="fa-solid fa-xmark"></i>
            </button>
        </td>
    </tr>
    `;
    cartTable.innerHTML += row;
});

if (totalPriceElement) {
    totalPriceElement.textContent = `$$${totalPrice}`;
}
updateBadge();
}

```

У лістингу 3.7 представлено дві ключові функції, що забезпечують синхронізацію внутрішнього стану даних із їх візуальним представленням. Функція `updateBadge()` відповідає за управління числовим індикатором, розташованим на кінці кошика в навігаційній панелі. Алгоритм функції передбачає ітерацію масиву `cart` за допомогою методу `forEach` для акумуляції загальної кількості одиниць продукції. Залежно від отриманого результату, властивість `display` елемента змінюється програмно, що дозволяє приховувати порожній індикатор та відображати його лише при наявності товарів.

Функція `renderCart()` реалізує повний цикл оновлення вмісту виїзної панелі кошика. На першому етапі властивість `innerHTML` контейнера `cartTable` скидається до порожнього рядка, що необхідно для запобігання дублюванню рядків при повторних викликах [27]. Далі в циклі виконується математичний розрахунок проміжної вартості для кожної позиції (`productTotal`) та накопичення загальної суми замовлення у змінній `totalPrice`.

Особливістю реалізації є використання шаблонних рядків JavaScript для динамічної генерації HTML-структури таблиці. Кожен згенерований рядок містить кнопку видалення з унікальним атрибутом `data-id`, що забезпечує зворотний зв'язок із логікою видалення товарів.

Після завершення ітерації функція оновлює текстовий вміст елемента `totalPriceElement`, виводячи підсумкову суму, та викликає `updateBadge()`. Такий ланцюговий виклик функцій гарантує, що будь-яка зміна у складі кошика буде

миттєво та комплексно відображена в усіх частинах інтерфейсу вебдодатка, забезпечуючи цілісність сприйняття системи користувачем.

3.2.4 Робота з Offcanvas-компонентом та подіями

Для забезпечення сучасного та динамічного користувацького інтерфейсу в проєкті було відмовлено від використання окремої сторінки кошика на користь компонента Offcanvas (виїзна бічна панель) фреймворку Bootstrap 5. Таке рішення дозволяє користувачеві взаємодіяти з обраними товарами, не залишаючи сторінку каталогу, що суттєво покращує показники утримання та зручність навігації. Керування станом цієї панелі та обробка навігаційних подій реалізовані за допомогою об'єктної моделі подій JavaScript.

Програмний код, що відповідає за ініціалізацію та керування візуальним станом кошика, наведено у лістингу 3.8.

Лістинг 3.8 – Керування інтерфейсом кошика та обробка подій

```
document.getElementById('cart-open').addEventListener('click',
(e) => {
    e.preventDefault();
    offcanvasCart.toggle();
});

document.querySelectorAll('.closecart').forEach(item => {
    item.addEventListener('click', (e) => {
        e.preventDefault();
        offcanvasCart.hide();
        let href = item.dataset.href;
        if (document.getElementById(href)) {
            document.getElementById(href).scrollIntoView({
behavior: 'smooth' });
        }
    });
});
```

У лістингу 3.8 продемонстровано механізм взаємодії з методами екземпляра класу Offcanvas. Обробник події для елемента з ідентифікатором cart-open використовує метод e.preventDefault(), який блокує стандартну поведінку браузера (перехід за посиланням або оновлення сторінки), дозволяючи скрипту самостійно

керувати логікою відображення. Виклик методу `toggle()` забезпечує динамічне перемикання стану панелі між видимим та прихованим [28].

За допомогою методу `querySelectorAll` система знаходить усі елементи з класом `.closecart` та призначає їм слухачі подій через цикл `forEach`. Це дозволяє реалізувати складний сценарій: при кліку на посилання всередині кошика (наприклад, перехід до певного розділу каталогу) панель автоматично закривається за допомогою методу `hide()`.

Після приховування панелі скрипт зчитує значення атрибута `data-href` (ідентифікатор цільового блоку) та використовує метод `scrollIntoView()`. Параметр `behavior: 'smooth'` забезпечує плавне вертикальне прокручування сторінки до потрібного контенту, що створює ефект безперервної взаємодії та підвищує загальну ергономічність вебдодатка.

Така організація роботи з компонентами та подіями дозволяє мінімізувати когнітивне навантаження на користувача, роблячи процес керування покупками інтуїтивно зрозумілим та технічно стабільним.

3.3 Реалізація підсистеми авторизації та ідентифікації

Підсистема ідентифікації та авторизації є критично важливим компонентом будь-якого інтернет-магазину. Вона забезпечує персоналізацію користувацького досвіду, розмежування прав доступу до певних функцій (наприклад, оформлення замовлення доступне лише для авторизованих клієнтів) та надійну прив'язку історії транзакцій до конкретного профілю у базі даних. У розроблюваній системі реалізовано класичну модель автентифікації на основі збереження сесансу за допомогою механізму Cookie-файлів.

Базовим елементом для роботи всіх скриптів підсистеми є модуль підключення до бази даних, винесений в окремий файл `db.php`. Це дозволяє уникнути дублювання коду та забезпечує єдину точку конфігурації. У цьому

файлі ініціалізується об'єкт `mysqli` із вказівкою параметрів сервера, логіна, пароля та назви бази даних, а також встановлюється кодування `utf8mb4` для коректної обробки кирилических символів.

Процес створення нового облікового запису обробляється серверним скриптом `check.php`. Головним завданням цього етапу є не лише збереження даних у таблиці `users`, але й попередня валідація та очищення вхідної інформації для забезпечення безпеки системи.

Програмний код обробника реєстрації наведено у лістингу 3.9.

Лістинг 3.9 – PHP-скрипт реєстрації користувача

```
<?php
require_once 'db.php';
$login = filter_var(trim($_POST['login']), FILTER_SANITIZE_STRING);
$password = filter_var(trim($_POST['pass']), FILTER_SANITIZE_STRING);
if (mb_strlen($login) < 4 || mb_strlen($login) > 100) {
    echo "Недопустима довжина логіна";
    exit();
} else if (mb_strlen($password) < 2 || mb_strlen($password) > 32) {
    echo "Недопустима довжина пароля (від 2 до 32 символів)";
    exit();
}
$mysql->query("INSERT INTO `users` (`login`, `password`)
VALUES('$login', '$password')");
$mysql->close();
header('Location: register.php'); ?>
```

У лістингу 3.9 дані, отримані з глобального масиву `$_POST`, проходять обов'язкову санітизацію. Функція `trim()` видаляє зайві пробіли на початку та в кінці рядка, а `filter_var()` з параметром `FILTER_SANITIZE_STRING` вирізає потенційно небезпечні HTML-теги, що є базовим захистом від міжсайтового скриптингу (XSS).

За допомогою функції `mb_strlen()`, маємо можливість коректно рахувати символи у багатобайтових кодуваннях. Якщо дані відповідають встановленим критеріям безпеки (логін від 4 до 100 символів, пароль від 2 до 32), система формує SQL-запит `INSERT` для створення нового запису в базі даних. Після

успішного виконання транзакції з'єднання з БД закривається (`$mysql->close()`), а клієнт перенаправляється на сторінку реєстрації за допомогою функції `header()`.

Процес входу до системи (автентифікація) та ініціалізації сеансу реалізовано у скрипті `auth.php`. Цей скрипт звіряє введені користувачем облікові дані з тими, що зберігаються в базі, і у разі збігу генерує ідентифікатор сесії на стороні клієнта.

Код модуля автентифікації наведено у лістингу 3.10.

Лістинг 3.10 – PHP-скрипт автентифікації та ініціалізації сеансу

```
<?php
require_once 'db.php';

$login = filter_var(trim($_POST['login']), FILTER_SANITIZE_STRING);
$password = filter_var(trim($_POST['pass']), FILTER_SANITIZE_STRING);

if ($mysql->connect_error) {
    die("Помилка підключення до бази даних: " . $mysql->connect_error);
}

$result = $mysql->query("SELECT * FROM `users` WHERE `login` = '$login' AND `password` = '$password'");

if (!$result) {
    die("Помилка обробки запроса: " . $mysql->error);
}

$user = $result->fetch_assoc();

if ($user === null) {
    echo "Такий користувач не знайдено";
    exit;
}

if (!empty($user['login'])) {
    setcookie('user', $user['login'], time() + 3600, "/");
} else {
    echo "Ім'я користувача пусте.";
    exit;
}

$mysql->close();
header('Location: index.php');
?>
```

Після стандартної процедури підключення до БД та санітизації вхідних параметрів, скрипт формує SQL-запит SELECT для пошуку користувача із вказаною комбінацією логіна та пароля. Метод `fetch_assoc()` перетворює знайдений рядок бази даних на асоціативний масив. Якщо масив `$user` дорівнює `null`, це означає, що облікові дані невірні, і виконання скрипта зупиняється.

Ключовим механізмом збереження стану авторизованого користувача є використання функції `setcookie()`. У разі успішної автентифікації сервер відправляє браузеру спеціальний HTTP-заголовок, який наказує зберегти на пристрої клієнта куки-файл з іменем `user` та значенням логіна користувача. Параметр `time() + 3600` встановлює термін дії сеансу – 1 година з моменту входу. Параметр шлях `"/"` вказує на те, що цей куки-файл буде доступний на всіх сторінках вебдодатка. Після ініціалізації сеансу відбувається автоматичний редирект на головну сторінку каталогу.

Наявність Cookie-файлу `user` дозволяє клієнтській частині (через PHP-рендер) визначати поточний стан відвідувача та динамічно змінювати елементи навігаційної панелі. Це реалізовано безпосередньо у загальному шаблоні (шапці) сайту, як показано у лістингу 3.11.

Лістинг 3.11 – Динамічний рендеринг елементів інтерфейсу на основі стану сеансу

```
<div class="btn-group me-2">
  <?php if (!isset($_COOKIE['user'])): ?>
    <div class="dropdown">
      <button class="btn btn-md dropdown-toggle"
type="button" data-bs-toggle="dropdown" aria-expanded="false">
        Аккаунт
      </button>
      <ul class="dropdown-menu">
        <li><a class="dropdown-item"
href="login.php">Увійти</a></li>
        <li><a class="dropdown-item"
href="register.php">Зареєструватись</a></li>
      </ul>
    </div>
  <?php else: ?>
    <div>
      <span class="me-2"><?php echo
htmlspecialchars($_COOKIE['user']); ?></span>
```

```

        <a href="logout.php"class="btn btn-md btn-
don">Вийти</a>
    </div>
    <?php endif; ?>
</div>

```

В основі логіки візуалізації лежить перевірка за допомогою конструкції `if (!isset($_COOKIE['user']))`. Якщо кукі-файл відсутній (користувач є гостем), інтерфейс генерує кнопку «Аккаунт» із випадаючим меню для переходу на сторінки реєстрації та авторизації. Якщо ж сеанс активний, система виводить логін користувача та відображає кнопку «Вийти».

Функція виходу із системи делегована окремому скрипту. Його логіка максимально мінімалістична, але ефективна:

Лістинг 3.12 – Функція виходу із системи

```

<?php
setcookie('user', '', time() - 3600, "/");
header('Location: index.php');
exit();
?>

```

Для знищення активного сеансу функція `setcookie()` викликається з ідентичним ключем `user`, але з порожнім значенням та часом життя, встановленим у минулому (`time() - 3600`). Це змушує браузер клієнта негайно видалити кукі-файл з локальної пам'яті. Після цього відбувається редирект на головну сторінку, і інтерфейс автоматично повертається до стану неавторизованого гостя.

3.4 Розробка модуля оформлення замовлення

Модуль оформлення замовлення виконує функцію об'єднання персональних даних з форми доставки та масивів обраних товарів із кошика, а потім безпечно передати їх на сервер для збереження в базі даних. Для

забезпечення безперервного користувацького досвіду та уникнення перезавантаження сторінки під час транзакції, передача даних реалізована за принципами AJAX з використанням сучасного інтерфейсу Fetch API.

Програмна реалізація клієнтської частини обробки замовлення наведена у лістингу 3.12.

Лістинг 3.12 – JavaScript-скрипт формування та відправки замовлення

```
const cart = JSON.parse(localStorage.getItem('chitankaCart'))
|| [];
if (cart.length === 0) {
    alert("Спочатку додайте книги до кошика!");
    return;
}
const orderData = {
    name: document.getElementById('name').value,
    email: document.getElementById('email').value,
    phone: document.getElementById('phone').value,
    city: document.getElementById('city').value,
    address: document.getElementById('address').value,
    payment_method:
document.querySelector('input[name="payment-
method"]:checked').value || 'Не вказано',
    total_price: parseFloat(document.getElementById('checkout-
total').textContent),
    cart: cart
};
fetch('save_order.php', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(orderData)
})
.then(res => res.json())
.then(data => {
    if (data.success) {
        localStorage.removeItem('chitankaCart');
        alert('Дякуємо! Замовлення буде відправлено!');
        window.location.href = 'index.php';
    } else {
        alert('Помилка: ' + data.message);
    }
})
.catch(err => alert('Помилка з\'єднання з сервером.'));
```

На початку виконання скрипта у лістингу 3.12 відбувається обов'язкова перевірка стану локального сховища. Якщо кошик порожній, виконання функції

переривається, що виключає можливість відправки «порожніх» транзакцій. Далі формується комплексний об'єкт `orderData`, який об'єднує дані текстових полів форми, вибраний метод оплати (через пошук активованої радіо-кнопки), загальну суму та масив об'єктів товарів `cart`.

Метод `fetch()` ініціює асинхронний POST-запит до серверного обробника `save_order.php`. Заголовок `Content-Type: application/json` чітко вказує серверу формат переданої інформації, а сам об'єкт перетворюється на рядок за допомогою `JSON.stringify()`. Завдяки використанню ланцюжка промісів (`.then()`), система очікує на відповідь сервера. Якщо транзакція пройшла успішно (`data.success`), скрипт автоматично очищує локальне сховище користувача (`localStorage.removeItem`), виводить повідомлення про успіх та перенаправляє клієнта на головну сторінку.

Серверна частина модуля оформлення замовлення має алгоритм, який включає зчитування вхідного JSON-потoku, перевірку сеансу користувача, санітизацію даних та виконання складеного транзакційного запису до реляційних таблиць бази даних.

Логіка збереження замовлення у системі управління базами даних MySQL наведена у лістингу 3.13.

Лістинг 3.13 – PHP-скрипт транзакційної обробки замовлення

```
<?php
require_once 'db.php';

$data = json_decode(file_get_contents('php://input'), true);

$user_login = $_COOKIE['user'] ?? null;
$user_id = "NULL";
if ($user_login) {
    $user_login = $mysql->real_escape_string($user_login);
    $user_find = $mysql->query("SELECT `id` FROM `users` WHERE
`login` = '$user_login'");
    if ($user_data = $user_find->fetch_assoc()) {
        $user_id = (int)$user_data['id'];
    }
}

$name = $mysql->real_escape_string($data['name']);
$email = $mysql->real_escape_string($data['email']);
```

```

$total_price = (float)$data['total_price'];
$cart = $data['cart'];

$query_order = "INSERT INTO `orders` (`user_id`, `name`,
`email`, `total_price`, `created_at`)
VALUES ($user_id, '$name', '$email',
'$total_price', NOW())";

if ($mysql->query($query_order)) {

    $order_id = $mysql->insert_id;

    foreach ($cart as $item) {
        $book_id = (int)$item['id'];
        $quantity = (int)$item['quantity'];
        $price = (float)$item['price'];

        $mysql->query("INSERT INTO `order_items` (`order_id`,
`book_id`, `quantity`, `price`)
VALUES ($order_id, $book_id, $quantity,
$price)");
    }
    echo json_encode(['success' => true]);
} else {
    echo json_encode(['success' => false, 'message' => 'Помилка
бази даних']);
}
?>

```

Процес серверної обробки у лістингу 3.13 починається зі зчитування вхідного потоку даних за допомогою конструкції `file_get_contents('php://input')` та його декодування у PHP-масив. Наступним кроком є ідентифікація користувача: скрипт перевіряє наявність сесійного Cookie-файлу, екранує його вміст та робить запит до таблиці `users`, щоб отримати унікальний числовий ідентифікатор клієнта (`user_id`). Якщо користувач не авторизований, системна логіка може відхилити запит або (у поточній конфігурації) записати його як гостя. Далі всі текстові поля проходять обов'язкову санітизацію через метод `$mysql->real_escape_string()`, що запобігає ризикам SQL-ін'єкцій під час виконання запитів до БД.

Архітектура збереження даних розроблена за правилами нормалізації реляційних баз даних і складається з двох етапів. На першому етапі формується

запит INSERT до таблиці orders, куди записується "шапка" замовлення: ідентифікатор користувача, контактна інформація та загальна сума. Після успішного виконання цього запиту викликається вбудована властивість `$mysql->insert_id`, яка повертає унікальний первинний ключ щойно створеного замовлення. На другому етапі отриманий ключ (`$order_id`) використовується як зовнішній ключ (Foreign Key) у циклі `foreach`. Система ітерує масив придбаних книг (`$cart`) та для кожної позиції формує окремий запит до підлеглої таблиці `order_items`. Завдяки цьому кожна книга жорстко прив'язується до конкретного замовлення. Після успішного завершення всіх транзакцій сервер формує та відправляє клієнту відповідь у форматі JSON (`{"success": true}`), яка є сигналом для клієнтського скрипта про успішне завершення покупки.

3.5 Висновок до розділу 3

У третьому розділі було виконано програмну реалізацію всіх клієнтських та серверних компонентів комп'ютерної системи інтернет-магазину «CHITANKA» на основі обраного стеку технологій. Розробку користувацького інтерфейсу (фронтенду) реалізовано за допомогою стандартів HTML5, каскадних таблиць стилів CSS3 та інструментарію фреймворку Bootstrap 5. Це дозволило створити гнучку, адаптивну модульну сітку, яка безпомилково адаптується під екрани мобільних пристроїв та десктопів, забезпечуючи високу ергономічність. Інтерактивні функції каталогу, зокрема механізм миттєвої фільтрації книг за категоріями через маніпуляції з DOM-деревом властивостей, а також клієнтська логіка кошика побудовані на базі чистого JavaScript. Впровадження технології Web Storage API (LocalStorage) дозволило забезпечити автономне збереження стану обраних товарів між сесіями користувача та реалізувати реактивний перерахунок вартості замовлення в реальному часі без надлишкових запитів до сервера.

Серверну частину (бекенд) спроектовано мовою програмування PHP з використанням розширення MySQLi для безпечної взаємодії з базою даних. Розроблено модулі реєстрації та автентифікації користувачів, які підтримують захищене збереження сесій на основі механізму Cookie-файлів. З метою захисту від поширених вразливостей (SQL-ін'єкцій та атак типу XSS) реалізовано обов'язкову первинну санітизацію, екранування та фільтрацію всіх вхідних POST-параметрів. Обробку фінальної стадії замовлення організовано за допомогою асинхронних запитів Fetch API, завдяки чому серверний скрипт `save_order.php` успішно виконує атомарний двостадійний запис заголовка транзакції та деталізованої специфікації куплених книг у пов'язані таблиці баз даних, гарантуючи швидкість та надійність операцій.

4 ВИПРОБУВАННЯ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Випробування інтерфейсу користувача

Першим етапом випробування сайту книжкового магазину став користувацький інтерфейсу (UI) на предмет адаптивності та ергономічності. Оскільки у електронній комерції значна частина користувачів здійснює покупки з мобільних пристроїв, критично важливою вимогою було забезпечення коректного масштабування та перебудови контенту під різні роздільні здатності екранів.

Для перевірки чуйності дизайну (Responsive Web Design), реалізованого за допомогою системи сіток фреймворку Bootstrap 5, була проведена ручна перевірка. Випробування здійснювалося як у стандартному вікні браузера для персональних комп'ютерів, так і з використанням інструментів розробника (Developer Tools) для симуляції відображення на мобільних пристроях різних форм-факторів.

Як показало випробування десктопної версії (рис. 4.1), на широких екранах інтерфейс головної сторінки та каталогу товарів використовує доступну ширину

максимально ефективно. Картки книг вибудовуються у класичну сітку, що забезпечує високу інформативність. Головна навігаційна панель повністю розгорнута, надаючи швидкий доступ до керування акаунтом, вибору мови та інтерактивного кошика.

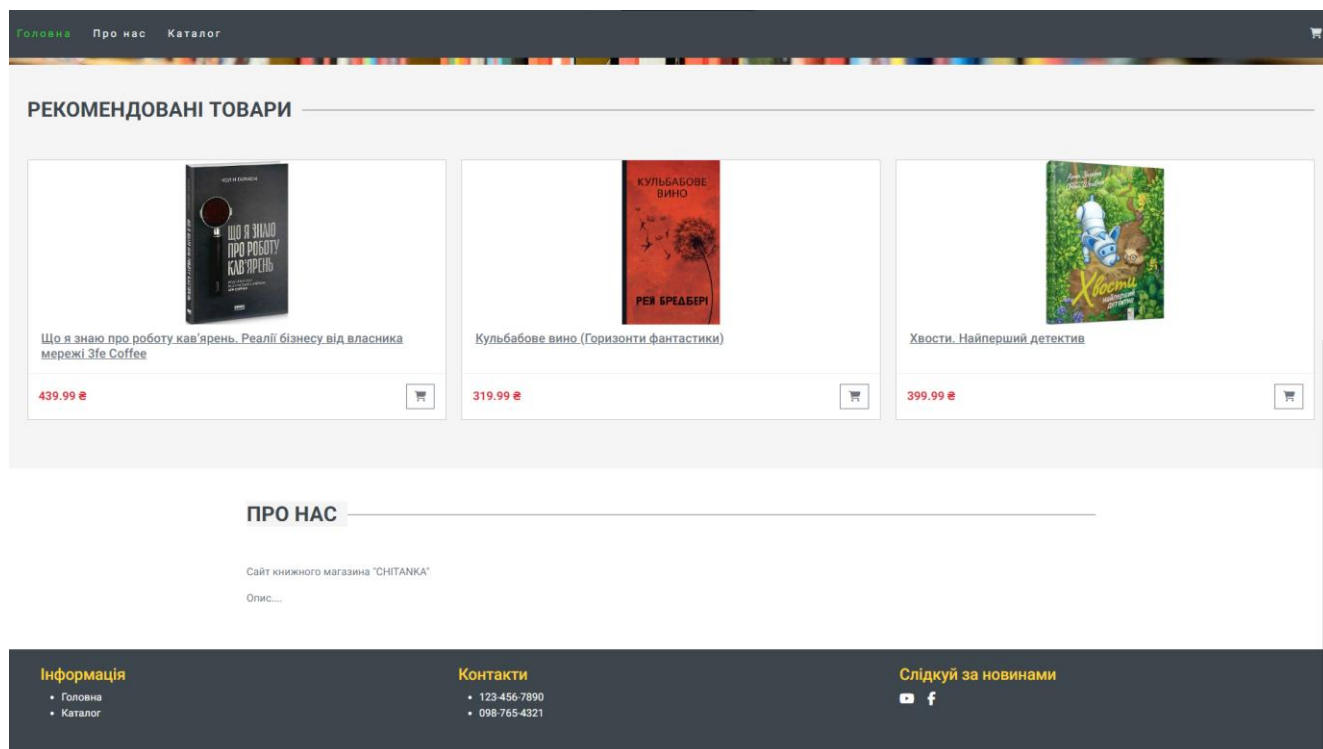


Рисунок 4.1 – Результат перевірки на ПК

Випробування також було приділено тестуванню на вузьких екранах смартфонів. Під час симуляції відображення на пристрої Pixel 7 (рис. 4.2) система успішно продемонструвала автоматичну перебудову архітектури сторінки. Спрацювали відповідні медіа-запити (breakpoints):

Шапка сайту (Header) реорганізувалася за допомогою класів порядковості (order), вивівши логотип та контактну інформацію у зручний для мобільного користувача формат без візуального перекриття елементів.

Сітка товарів трансформувалася у представлення по одному елементу в ряд. Це гарантує збереження читабельності описів, чітке відображення цін та достатній розмір кнопок «Додати до кошика» для комфортного натискання сенсорним екраном (touch-friendly).

Основне меню навігації та панель фільтрів приховалися у компактний виїзний блок (Offcanvas), заощаджуючи робочий простір екрана.

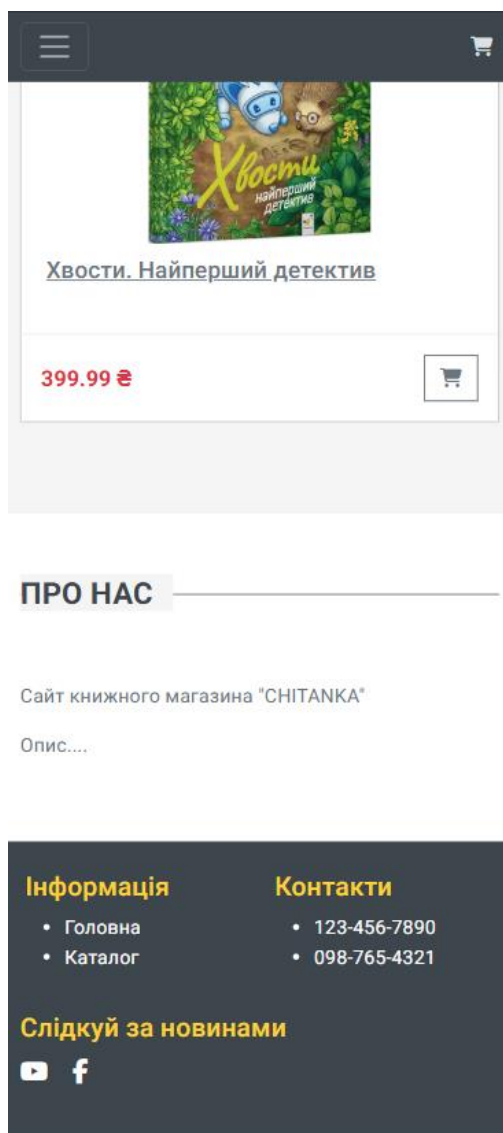


Рисунок 4.2 – Результат перевірки на пристрою Pixel 7

Було перевірено відображення системи на екранах середнього розміру та нестандартних форматах, зокрема на пристрої Asus Zenbook Fold (рис. 4.3). На цьому етапі перевірка підтвердила, що контент рівномірно розподіляється по площі екрана, зображення обкладинок книг масштабуються пропорційно, а всі інтерактивні елементи залишаються доступними без необхідності горизонтального прокручування (scroll).

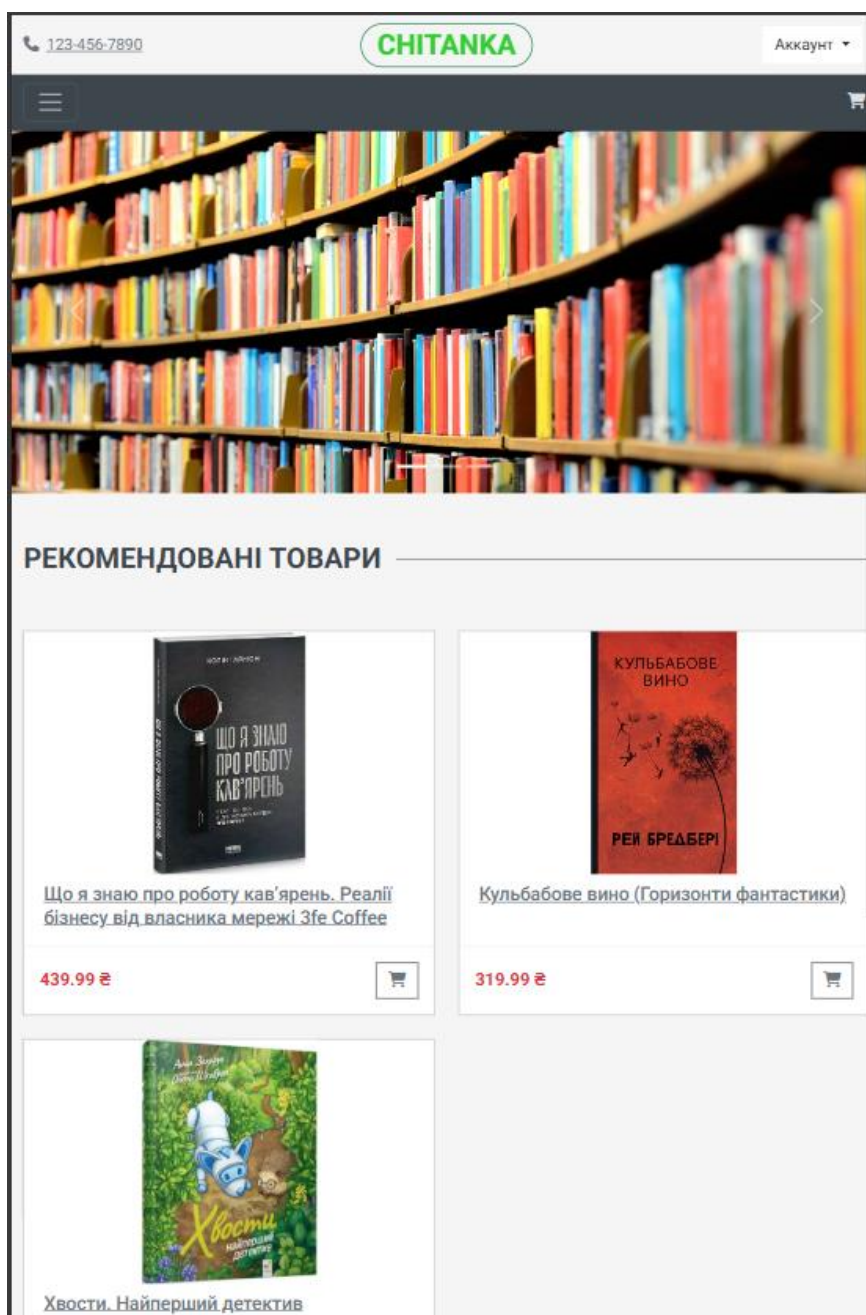


Рисунок 4.3 – Результат перевірки на пристрою Asus Zenbook Fold

4.2 Перевірка механізмів навігації та фільтрації товарів

Після перевірки візуального відображення було проведено випробування функціональних можливостей каталогу, а саме механізму динамічного сортування книг за категоріями. Метою цього етапу було підтвердження коректності роботи

клієнтської логіки, яка дозволяє користувачеві швидко знаходити потрібну літературу без повного оновлення вебсторінки. Для прикладу на рисунку 4.4 додані книги без обраних категорій.

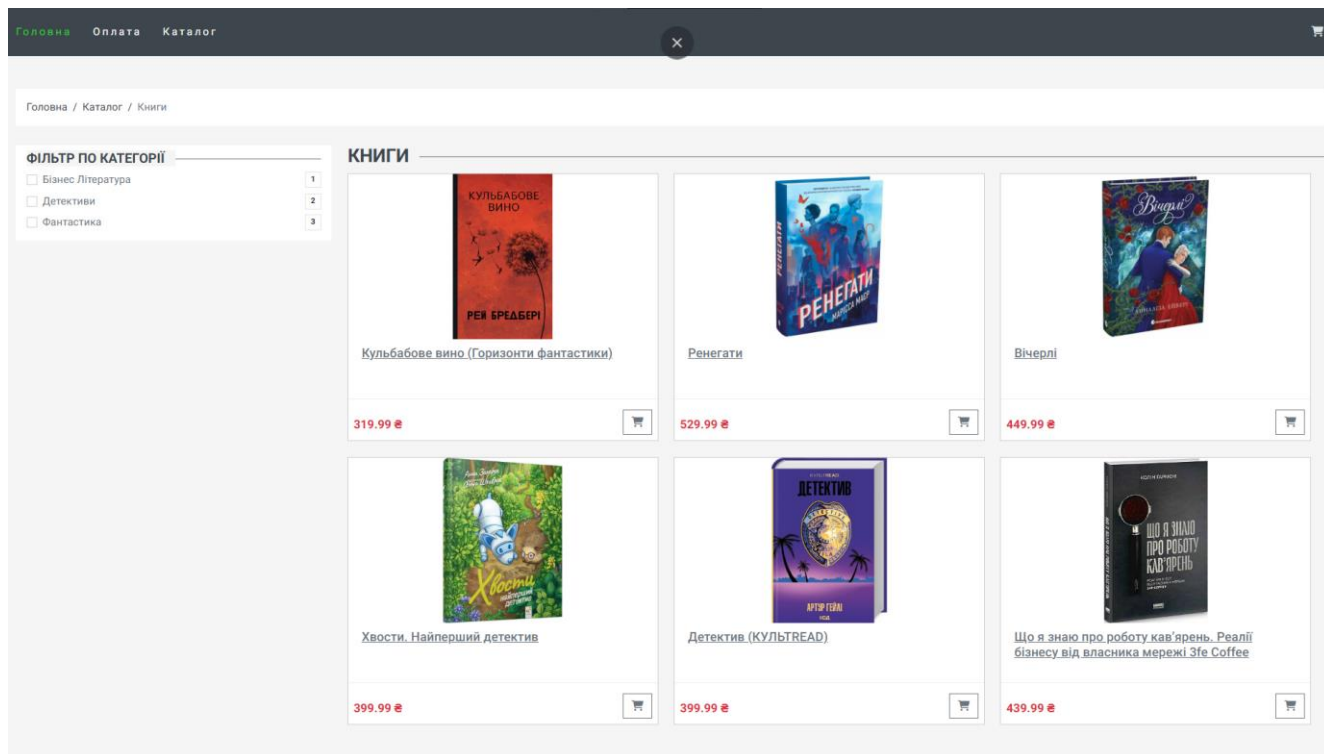


Рисунок 4.4 – Вивід книг каталогу без фільтра

Випробування проводилося шляхом активації різних фільтрів у бічній панелі (Sidebar). Як продемонстровано на рисунку 4.5, при виборі категорії «Бізнес Література», та на рисунку 4.6 при виборі декількох категорій, JavaScript-сценарій миттєво ідентифікує стан відповідних чекбоксів та виконує ітерацію за масивом товарних карток.

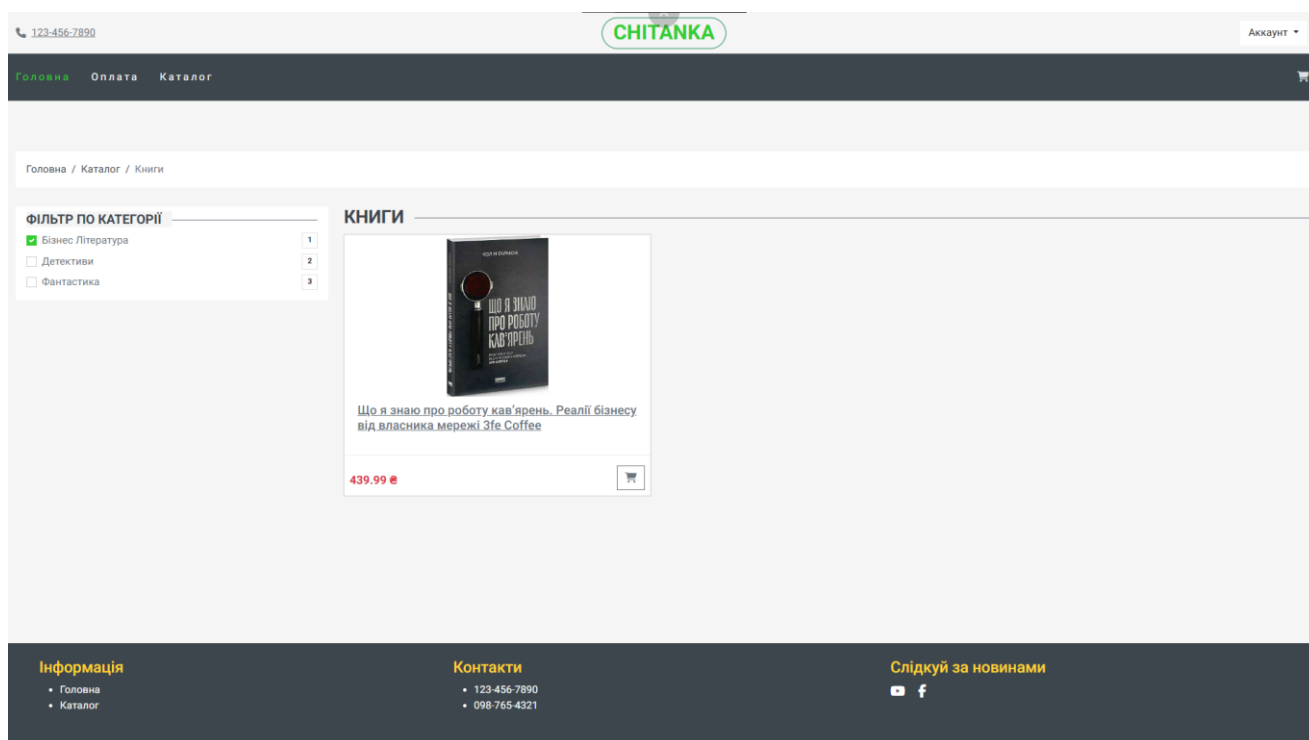


Рисунок 4.5 – Вивід книг каталогу з фільтром «Бізнес Література»

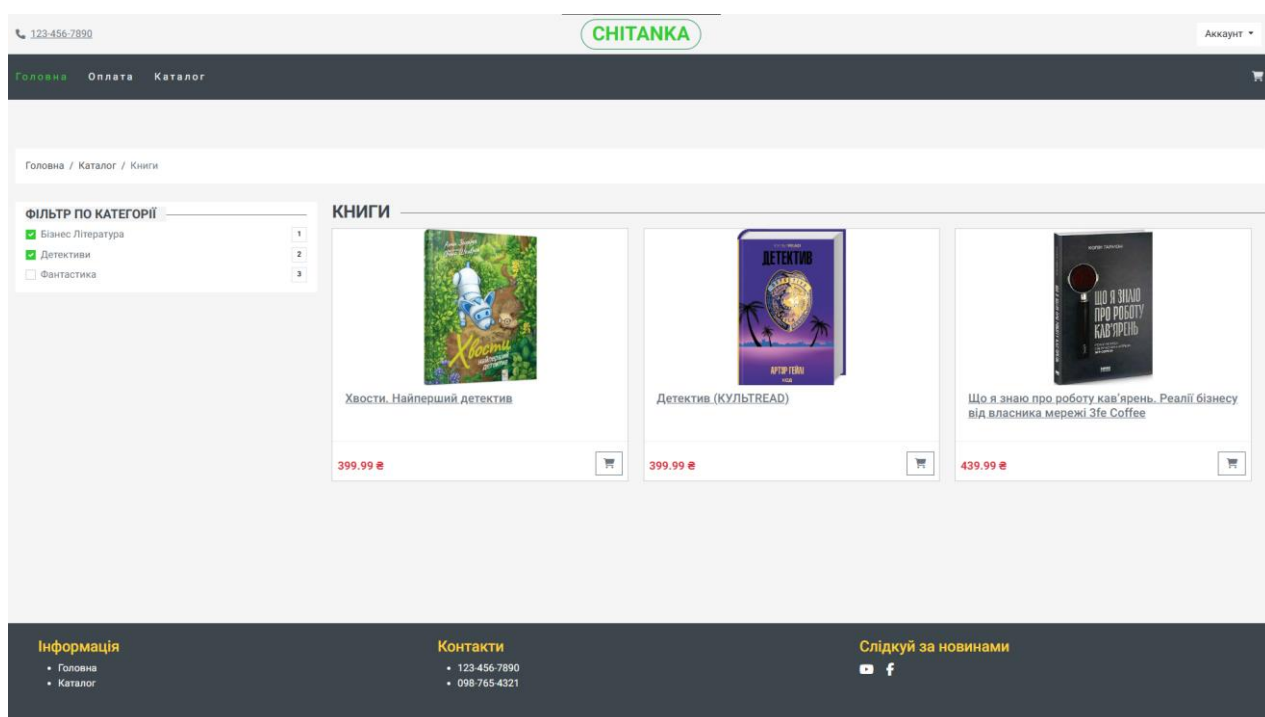


Рисунок 4.6 – Вивід книг каталогу з фільтром «Бізнес Література» та «Детективи»

Під час випробувань було перевірено кілька сценаріїв взаємодії:

Вибір однієї категорії: система успішно приховує всі книги, чий атрибут data-category не збігається з ідентифікатором обраного фільтра.

Мультифільтрація: при виборі одночасно двох або більше категорій (наприклад, «Детективи» та «Бізнес») каталог відображає об'єднаний набір товарів, що відповідають хоча б одній із вибраних умов.

Скидання фільтрів: при знятті всіх позначок функція `filterProducts` повертає нульову довжину масиву обраних категорій, що ініціює відображення всього асортименту магазину.

Також було перевірено роботу навігаційних посилань у футері та меню. Перевірка підтвердила, що всі внутрішні переходи працюють коректно, забезпечуючи цілісну структуру сайту.

4.3 Перевірка роботи кошика та динамічного перерахунку

На етапі випробувань функціональності інтерактивного кошика та коректності роботи механізмів клієнтської логіки, основною метою полягало у підтвердженні того, що додавання, видалення товарів та перерахунок загальної вартості відбуваються динамічно, без затримок та потреби у перезавантаженні сторінок.

Під час перевірки операції додавання книг до кошика було підтверджено працездатність системи візуальних індикаторів. Як продемонстровано на скриншотах у роботі (рис. 4.7), додамо 3 товара натиснувши кнопку кошика, числовий показник на бейджі у верхній навігаційній панелі миттєво оновлюється. Це підтверджує коректність роботи функції `updateBadge()`, яка в реальному часі підсумовує кількість одиниць усіх об'єктів у масиві та змінює стан видимості елемента залежно від наявності товарів.

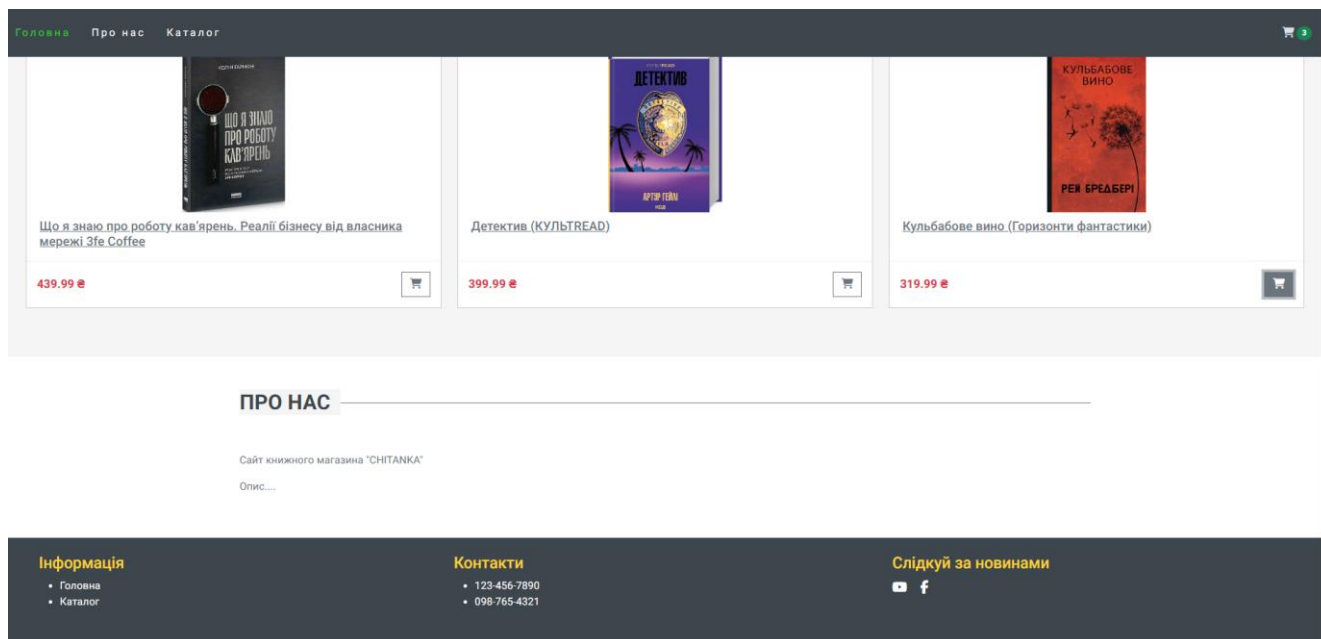


Рисунок 4.7 – Додавання товару до кошика

Детальний аналіз вмісту кошика проводився за допомогою виїзної панелі Offcanvas. Перевірка показала, що функція `renderCart()` безпомилково генерує таблицю обраних товарів (рис. 4.8). Для кожної позиції правильно відображаються назва, індивідуальна ціна та кількість книг. Також було приділено перевірку перевірки математичних обчислень: система коректно виконує множення ціни на кількість для кожного рядка та виводить підсумкову суму в нижній частині таблиці.

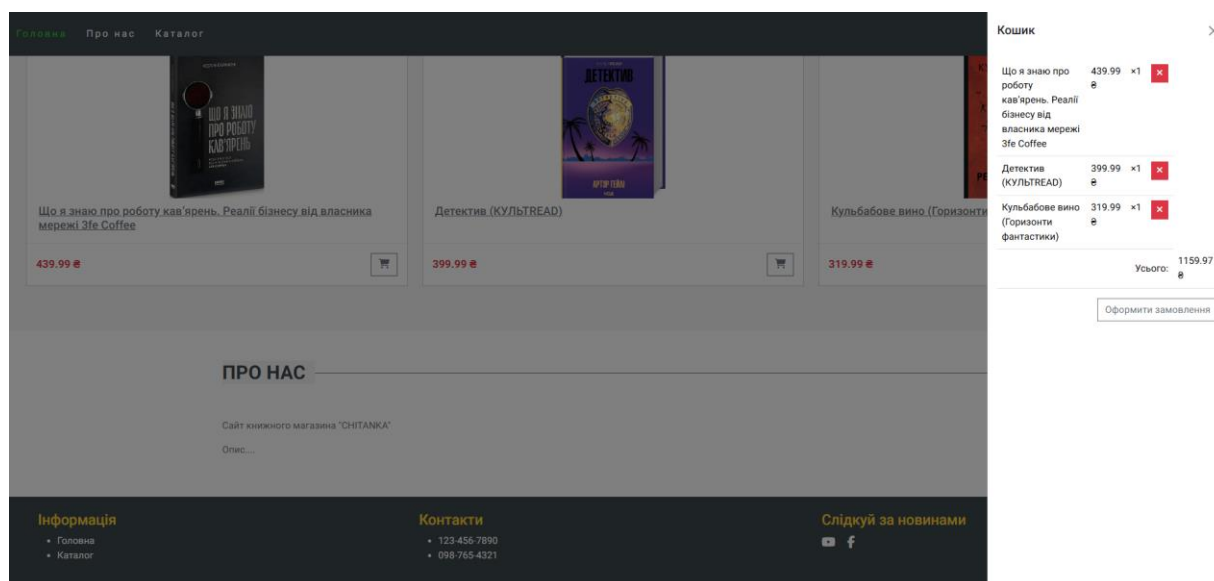


Рисунок 4.8 – Вивід доданих товарів у кошику

Перевіримо збереження стану кошика. Після додавання кількох книг було виконано перехід на іншу сторінку сайту (рис. 4.9). Завдяки реалізованому механізму Web Storage API, усі дані в кошику залишилися незмінними. Це доводить, що збереження даних у формат JSON та їх подальше відновлення при ініціалізації додатка працюють стабільно.

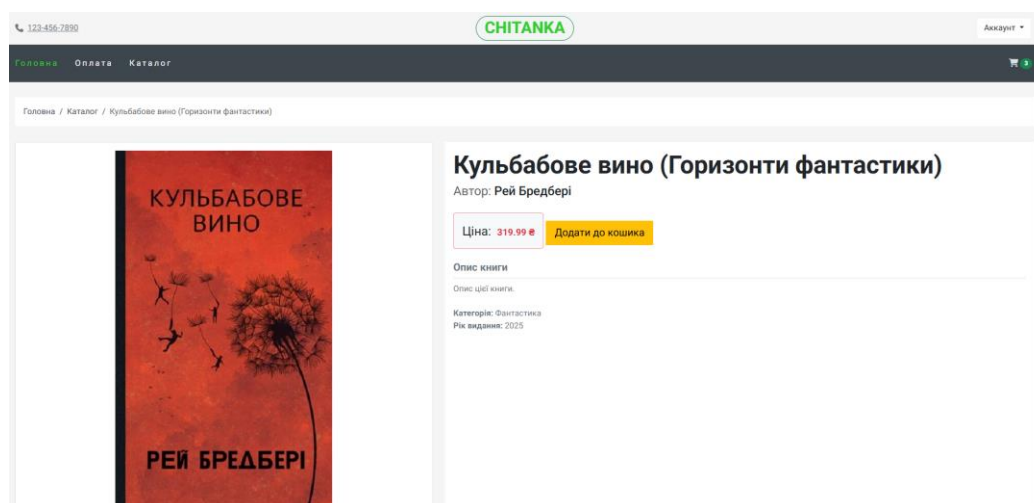


Рисунок 4.9 – Збереження кошика при переходу на іншу сторінку

4.4 Валідація даних при оформленні замовлення

Метою даного етапу випробувань є перевірка стійкості системи до некоректних дій користувача та підтвердження надійності механізмів захисту даних на сторінці оформлення замовлення. Випробування охоплювало перевірку прав доступу, валідацію вмісту кошика та коректність заповнення полів форми доставки.

Першочергово було перевірено механізм обмеження доступу для неавторизованих відвідувачів. При спробі прямого переходу на сторінку оформлення замовлення без активного сеансу (відсутність Cookie-файлу user) серверний сценарій PHP успішно ідентифікує відсутність прав доступу та виконує

автоматичне перенаправлення користувача на сторінку авторизації. Це гарантує, що персональні дані та процес транзакції залишаються захищеними.

Наступним кроком було протестовано захист від відправки «порожніх» замовлень. У разі, якщо користувач переходить до оформлення з порожнім кошиком, клієнтський JavaScript-сценарій блокує подальшу обробку запиту. Як продемонстровано на відповідному скриншоті (рис. 4.10), система виводить модальне сповіщення з текстом «Спочатку додайте книги до кошика!», що запобігає створенню некоректних записів у базі даних.

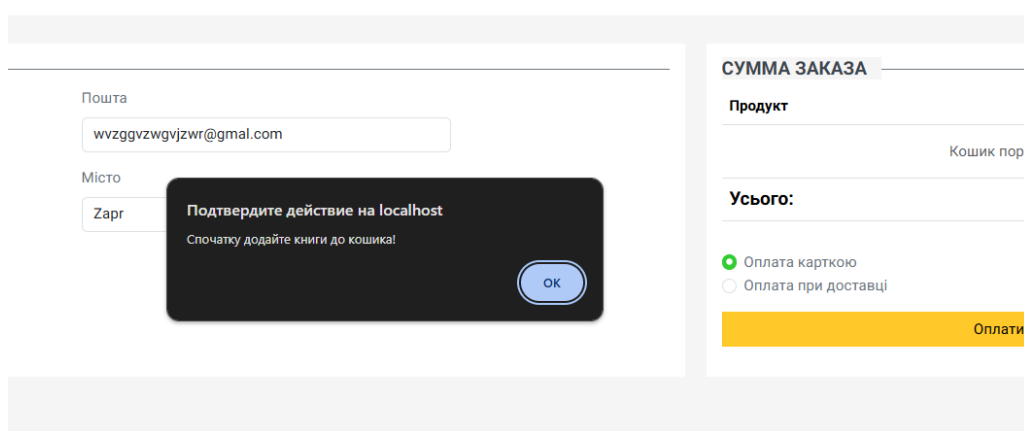


Рисунок 4.10 – Спроба оплати без товару

Під час випробування було імітовано відправку форми із незаповненими обов'язковими полями (прізвище, номер телефону, адреса). Завдяки використанню методу `checkValidity()` та класу `.needs-validation`, система автоматично заблокувала відправку запиту, вивівши підказки щодо необхідності їх заповнення (рис. 4.11).

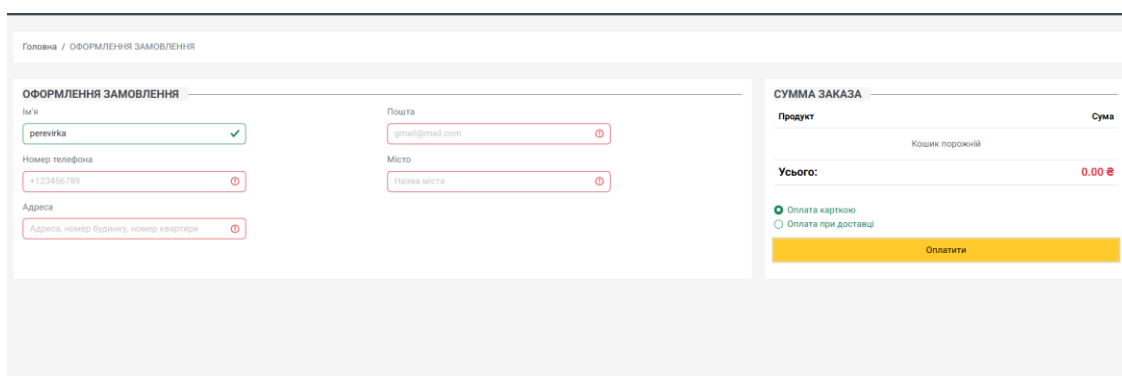


Рисунок 4.11 – Незаповнена форма замовлення

4.5 Перевірка серверної логіки та цілісності бази даних

У цьому етапі буде підтвердження того, що дані, передані з клієнтської частини через Fetch API, коректно обробляються PHP-скриптами та надійно зберігаються у відповідних таблицях бази даних із дотриманням усіх реляційних зв'язків.

Для проведення перевірки було виконано повний цикл оформлення замовлення авторизованим користувачем. Після заповнення форми на оформлення замовлення (рис. 4.12) та успішного відправлення запиту було проведено аудит бази даних за допомогою інструменту phpMyAdmin.

The screenshot displays the 'ОФОРМЛЕННЯ ЗАМОВЛЕННЯ' (Order Form) on the CHITANKA website. The form includes fields for Name, Email, Phone Number, Address, City, and Zip Code. A modal dialog is open, asking for confirmation to proceed on localhost. The right sidebar shows the 'СУММА ЗАКАЗА' (Order Total) with a list of items and their prices, and a total of 839.98. The bottom of the page contains navigation links and contact information.

Продукт	Сума
Що я знаю про роботу кав'ярень. Реаліі бізнесу від власника мережі 3fe Coffee x1	439.99 ₪
Детектив (КУЛЬТРЕАД) x1	399.99 ₪
Усього:	839.98 ₪

Рисунок 4.12 – Оформлення замовлення

Як продемонстровано на скриншоті (Рисунок 4.13), у таблиці orders було створено новий запис, що містить повну «шапку» замовлення: унікальний ідентифікатор користувача (user_id), контактні дані отримувача, метод оплати та підсумкову вартість. Поле created_at автоматично зафіксувало точний час здійснення транзакції.

Extra options

+ - +			id	user_id	name	email	phone	city	address	total_price	created_at	payment_method				
☐		Изменить		Копировать		Удалить	1	0	Вадим	vstupvadam@gmail.com	+3805785668867	Запорожье	Запорожское 6	1199.97	2026-04-11 19:27:22	Оплата при доставк
☐		Изменить		Копировать		Удалить	2	0	Tik	vstupvadam@gmail.com	+3805786=7098	Запорожье	Запорожское 2	1239.97	2026-04-11 19:28:25	Оплата при доставк
☐		Изменить		Копировать		Удалить	3	NULL	куцйк	vstupvadam@gmail.com	+769876978	Запорожье	Запорожское	2439.94	2026-04-11 19:34:07	Оплата картоку
☐		Изменить		Копировать		Удалить	4	NULL	іфавіф	vstupvadam@gmail.com	0537463764	Запорожье	Запорожское	0.00	2026-04-11 19:35:26	Оплата картоку
☐		Изменить		Копировать		Удалить	5	8	Киц	vstupvadam@gmail.com	+38088005959595	Запорожье	Запорожское 5	1299.97	2026-04-11 19:46:09	Оплата картоку
☐		Изменить		Копировать		Удалить	6	9	jkm	ese@gmail.com	334455	укуку	yhjkyuky	849.98	2026-04-11 21:32:43	Оплата при доставк
☐		Изменить		Копировать		Удалить	7	9	ольга	zxxzc@gmail.com	+386665544	киев	украинская 8-5	1969.95	2026-04-11 21:36:56	Оплата при доставк
☐		Изменить		Копировать		Удалить	8	10	dondon	wvzgvzvgvjzwr@gmail.com	+3805785758787	Зап	ZP	2079.94	2026-04-27 18:12:15	Оплата картоку
☐		Изменить		Копировать		Удалить	9	8	perevirka	wvzgvzvgvjzwr@gmail.com	+380909990990	Запг	ZP	839.98	2026-05-12 15:34:29	Оплата картоку

↑

☐ Отметить все

С отмеченными:

Изменить

Копировать

Удалить

Экспорт

Рисунок 4.13 – Таблица orders у MySQL

При перевірці цілісності у таблиці order_items (рис. 4.14), перевірка підтвердило, що для кожного товару з кошика було створено окремий рядок, де зовнішній ключ order_id чітко відповідає первинному ключу з таблиці замовлень. Це доводить коректність роботи серверної логіки щодо отримання insert_id та проведення циклічного запису складових частин замовлення.

Extra options										
				id	order_id	book_id	quantity	price		
☐		Изменить		Копировать		Удалить	1	1	4	3 399.99
☐		Изменить		Копировать		Удалить	2	2	5	1 399.99
☐		Изменить		Копировать		Удалить	3	2	4	1 399.99
☐		Изменить		Копировать		Удалить	4	2	6	1 439.99
☐		Изменить		Копировать		Удалить	5	3	6	1 439.99
☐		Изменить		Копировать		Удалить	6	3	4	2 399.99
☐		Изменить		Копировать		Удалить	7	3	5	3 399.99
☐		Изменить		Копировать		Удалить	8	5	1	1 319.99
☐		Изменить		Копировать		Удалить	9	5	2	1 529.99
☐		Изменить		Копировать		Удалить	10	5	3	1 449.99
☐		Изменить		Копировать		Удалить	11	6	1	1 319.99
☐		Изменить		Копировать		Удалить	12	6	2	1 529.99
☐		Изменить		Копировать		Удалить	13	7	5	2 399.99
☐		Изменить		Копировать		Удалить	14	7	1	2 319.99
☐		Изменить		Копировать		Удалить	15	7	2	1 529.99
☐		Изменить		Копировать		Удалить	16	8	1	4 319.99
☐		Изменить		Копировать		Удалить	17	8	4	2 399.99
☐		Изменить		Копировать		Удалить	18	9	6	1 439.99
☐		Изменить		Копировать		Удалить	19	9	5	1 399.99

Рисунок 4.14 – Таблица order_items у MySQL

Також при перевірці цілісності таблиці users (рис 4.15), було виявлено список користувачів яки були надіслані через форму реєстрації

	id	login	password
☐ Изменить ☐ Копировать ☐ Удалить	1	qrewqrew5234	321432253
☐ Изменить ☐ Копировать ☐ Удалить	3	vadik	667098798
☐ Изменить ☐ Копировать ☐ Удалить	4	Vadim	434531413431431
☐ Изменить ☐ Копировать ☐ Удалить	7	Qwerty	123
☐ Изменить ☐ Копировать ☐ Удалить	8	asdf	1234
☐ Изменить ☐ Копировать ☐ Удалить	10	mrdon	12345
☐ Изменить ☐ Копировать ☐ Удалить	11	asdf	123

Рисунок 4.15 – Таблица users у MySQL

Для узагальнення результатів проведених випробувань було складено таблицю функціональних характеристик, що відображає підсумки випробування ключових модулів системи (Таблиця 4.1)

Таблиця 4.1 – Функціональні характеристики Web-сайту книжкового магазину

№	Призначення	Дія	Отриманий результат	Підсумок
1	Перевірка на коректне додавання товарів у корзину	При натисканні на кнопку візка, товар додається до кошика	Під час натискання на кнопку товар додається в кошик.	Пройдено
2	Перевірка на коректну роботу фільтрів книг	При натисканні на один з фільтрів вибирається потрібна категорія	Коли натискають на один із фільтрів, обирається відповідна категорія.	Пройдено
3	Перевірка на коректне додавання до кошика числа книг при збільшенні числа	При збільшенні числа книг, що купуються, і додаванні їх до кошика, кількість книг, що купуються, відповідає числу книг у кошику	При збільшенні кількості придбаних книг і додаванні їх до кошика, число придбаних книг збігається з кількістю книг у кошику.	Пройдено
4	Перевірка на видання помилки при вході в аккаунт якого не існує	при наборі логіну і пароля якого ще немає в базі даних, пише помилку, що такого користувача немає	Якщо введені логін і пароль відсутні в базі даних, відображається повідомлення про помилку, що такого користувача не існує.	Пройдено
5	Перевірка на коректне створення та вхід в аккаунт	при реєстрації вводиться логін і пароль, після в вікні авторизації він вводиться і входить в обліковий запис який ми створили	Під час реєстрації користувач вводить логін і пароль, а потім використовує ці дані в вікні авторизації для входу до створеного облікового запису.	Пройдено

Аналіз даних таблиці 4.1 дозволяє стверджувати, що система демонструє стабільну роботу при виконанні основних бізнес-сценаріїв. Серверна логіка успішно обробляє вхідні потоки даних, що унеможлиблює появу ізольованих записів або некоректних фінансових показників.

4.6 Висновок до розділу 4

Четвертий розділ присвячено проведенню випробувань, верифікації та перевірці функціональної якості створеного програмного продукту. Тестування охоплювало аудит усіх рівнів архітектури вебдодатка: від перевірки адаптивності інтерфейсу до контролю транзакційної цілісності бази даних. Під час випробувань клієнтської логіки за допомогою інструментів розробника браузера було підтверджено коректність відпрацювання сценаріїв фільтрації та стабільність збереження стану інтерактивного кошика у локальному сховищі браузера, навіть при примусовому перезавантаженні сторінок (F5).

У ході тестування було приділено увагу підсистемі клієнтської та серверної валідації форм на сторінці оформлення замовлення. Система продемонструвала здатність блокувати некоректні дії користувача, успішно блокуючи спроби відправки порожніх або завідомо помилкових форм за допомогою модальних вікон-попереджень та візуальних Bootstrap-класів підсвічування. Перевірка серверної логіки та стану бази даних через середовище phpMyAdmin підтвердила, що кожна успішна покупка безперешкодно та синхронно реєструється у таблицях `orders` та `order_items` із точним логуванням часу транзакції (`created_at`) та збереженням реляційних зв'язків. Зведені результати випробувань, систематизовані у таблиці функціональних характеристик компонентів, доводять, що комп'ютерна система відповідає критеріям технічного завдання, працює стабільно та є готовою до практичного впровадження.

ВИСНОВОК

У результаті виконання дипломної роботи бакалавра було успішно вирішено актуальне завдання з проєктування, програмної реалізації та випробування сучасної розподіленої комп'ютерної системи інтернет-магазину книжкової продукції «CHITANKA». Виконана робота повністю відповідає поставленому технічному завданню та вимогам до сучасних систем електронної комерції.

На етапі аналітичного дослідження було проаналізовано стан ринку книготоргівлі в Україні та виявлено стійку тенденцію до цифровізації галузі. Дослідження існуючих аналогів дозволило сформулювати ключові вимоги до програмного продукту, серед яких головними визначено: висока швидкість роботи, адаптивність інтерфейсу під мобільні пристрої, безпека транзакцій та ергономічність взаємодії користувача з каталогом.

Під час етапу проєктування було обґрунтовано вибір класичної трирівневої клієнт-серверної архітектури (рівень представлення, рівень бізнес-логіки, рівень даних), що забезпечує високий рівень масштабованості та безпеки системи. За допомогою мови графічного моделювання UML розроблено діаграми прецедентів, діяльності та послідовностей, які дозволили чітко алгоритмізувати логіку роботи системи. Для забезпечення надійного зберігання інформації спроектовано реляційну базу даних у середовищі MySQL із дотриманням правил нормалізації.

Програмну реалізацію проєкту виконано з використанням сучасного стеку вебтехнологій: HTML5, CSS3, фреймворку Bootstrap 5 для клієнтського інтерфейсу, а також мов програмування JavaScript та PHP. У ході розробки реалізовані технічні рішення:

- створено адаптивний каталог товарів із механізмом асинхронної фільтрації за категоріями на стороні клієнта (JavaScript), що дозволяє миттєво сортувати контент без перезавантаження сторінки;

- розроблено інтерактивний кошик покупок із використанням Web Storage API (LocalStorage), що гарантує збереження обраних товарів між сесіями користувача та забезпечує динамічний перерахунок вартості замовлення;

- впроваджено надійну підсистему ідентифікації та авторизації користувачів на базі механізму Cookie-файлів із санітизацією вхідних даних для захисту від XSS та SQL-ін'єкцій;

- реалізовано транзакційний модуль оформлення замовлення (через Fetch API), який забезпечує цілісний двостадійний запис даних у пов'язані таблиці бази даних (orders та order_items).

Проведене комплексна перевірка системи (перевірка UI/UX, валідації форм, логіки кошика та цілісності записів у базі даних) підтвердило стабільну та безпомилкову роботу всіх модулів за різних сценаріїв використання.

Практичне значення та можливість впровадження: Розроблена програмна комп'ютерна система інтернет-магазину є готовою до практичного використання. Завдяки гнучкій архітектурі, спроектований інтернет-магазин може бути легко розгорнутий на будь-якому стандартному хостингу з підтримкою PHP/MySQL та впроваджений у реальні бізнес-процеси малого чи середнього підприємства книготоргівлі. Подальший розвиток системи може включати інтеграцію з платіжними шлюзами (API еквайрингу) та модулями служб доставки, що дозволить повністю автоматизувати цикл продажу, а також доробкам у напрямку оптимізації програмного коду, підвищення захищеності даних та інтеграції нових функціональних вебтехнологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шалева О. І. Електронна комерція: навч. посіб. Київ, 2011. 216 с.
2. Аналіз ринку книжкового видавничого ринку України. URL: <https://inventure.com.ua/uk/analytics/investments/analiz-rinku-knizhkovogo-vidavnichogo-rinku-ukrayini> (дата звернення 02.04.2026).
3. Кількість назв на книжковому ринку зростає, а накладі зменшуються – Український інститут книги. URL: <https://ubi.org.ua/uk/news/kategoriya-2/kilkist-nazv-na-knizhkovomu-rinku-zrosta-a-naklad-zmenshu-tsya-ukra-nskiy-institut-/knigi> (дата звернення 04.04.2026).
4. Круг С. Не змушуйте мене думати. Київ: Arthuss, 2024. 198 с.
5. «Книгарня Є»: офіційний сайт. URL: <https://book-ye.com.ua/> (дата звернення: 15.04.2026).
6. Ринок книг в Україні: одні мають мільйонні прибутки, інші працюють у мінус // Delo.ua: бізнес-портал. 2026. URL: <https://delo.ua/news/rinok-knig-v-ukrayini-odni-mayut-milionni-pributki-insi-pracyuyut-u-minus-462093/> (дата звернення: 15.04.2026).
7. Молчанов В. П., Пандорін О. К. Технології розробки WEB-ресурсів: навч. посіб. Харків: ХНЕУ ім. С. Кузнеця, 2019. 129 с.
8. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення: навч. посіб. Київ, 2018. 251 с.
9. Функціональні вимоги // Вікіпедія: вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Функціональні_вимоги (дата звернення: 15.04.2026).
10. Краус К. М. Електронна комерція та Інтернет-торгівля: навчально-методичний посібник. Київ: Аграр Медіа Груп, 2021. 456 с.
11. Баран С. В. Основи web-програмування: навчальний посібник. Кривий Ріг: Державний університет економіки і технологій, 2023. 316 с.
12. Каплун В. А., Ціхоцький М. С., Лукічов В. В. Основи WEB-програмування: навчальний посібник. Вінниця: ВНТУ, 2023. 129 с.

13. Нефункціональні вимоги // Вікіпедія: вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Нефункціональні_вимоги (дата звернення: 15.04.2026).

14. Веб-технології та веб-дизайн: навч. посіб. / Трофименко О. Г., Козін О. Б., Задерейко О. В., Плачінда О. Є. Одеса: Фенікс, 2019. 284 с.

15. Остапов С. Е., Євсєєв С. П., Король О. Г., Технології захисту інформації: навч. посіб. Харків: ХНЕУ ім. С. Кузнеця, 2013. 476 с.

16. Берко А. Ю., Верес О. М. Системи баз даних та знань: підручник. Львів: Магнолія 2006, 2021. 578 с.

17. Пасічник В. В., Пасічник О. В., Угрин Д. І. Веб-технології: підручник. Львів: Магнолія 2006, 2026. 334 с.

18. Bootstrap 5 Початок роботи // W3schoolsUA. URL: https://w3schoolsua.github.io/bootstrap5/bootstrap_get_started.html (дата звернення: 21.04.2026).

19. Грицюк Ю. І. Аналіз вимог до програмного забезпечення: навч. посіб. Львів: Видавництво Львівської політехніки, 2018. 456 с.

20. Мельник Р. А. Програмування веб-застосувань (фронт-енд та бек-енд): навч. посіб. Львів: Видавництво Львівської політехніки, 2018. 249 с.

21. Павловський В. І., Петрашенко А. В. Бази даних та засоби управління: підручник. Київ: КПП ім. Ігоря Сікорського, 2024. 293 с.

22. Поморова О. В., Говорущенко Т. О. Проектування інтерфейсів користувача: навч. посіб. Хмельницький: ХНУ, 2011. 206 с.

23. Ізмайлова О. В. Проектування інформаційних систем: навч. посіб. Київ: КНУБА, 2022. 87 с.

24. Кучеров Д. П., Артамонов Є. Б. Інженерія програмного забезпечення: навч. посіб. Київ: НАУ, 2017. 388 с.

25. Grid system // Bootstrap 5 Official Documentation. URL: <https://getbootstrap.com/docs/5.0/layout/grid/> (дата звернення: 01.05.2026).

26. Using data attributes // MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use_data_attributes (дата звернення: 04.05.2026).

27. Element: innerHTML property // MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML> (дата звернення: 04.05.2026).

28. Offcanvas via JavaScript // Bootstrap 5 Official Documentation. URL: <https://getbootstrap.com/docs/5.0/components/offcanvas/#via-javascript> (дата звернення: 08.05.2026).

ДОДАТОК А

ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 – Файл name.db.php

```
<?php
$mysql = new mysqli('localhost', 'root', '', 'vstupvadim');

if ($mysql->connect_error) {
    die("Помилка підключення до бази даних: " . $mysql->connect_error);
}
$mysql->set_charset("utf8mb4");
?>
```

Лістинг А.2 – Файл name.check.php

```
<?php
require_once 'db.php';
$login = filter_var(trim($_POST['login']),
FILTER_SANITIZE_STRING);
$password = filter_var(trim($_POST['pass']),
FILTER_SANITIZE_STRING);

if (mb_strlen($login) < 4 || mb_strlen($login) > 100) {
    echo "Недопустима довжина логіна";
    exit();
} else if (mb_strlen($password) < 2 || mb_strlen($password) >
32) {
    echo "Недопустима довжина пароля (від 2 до 32 символів)";
    exit();
}

$mysql->query("INSERT INTO `users` (`login`, `password`)
VALUES('$login', '$password')");

$mysql->close();

header('Location: register.php');
?>
```

Лістинг А.3 – Файл name.auth.php

```
<?php
require_once 'db.php';
```

```

    $login = filter_var(trim($_POST['login']),
FILTER_SANITIZE_STRING);
    $password = filter_var(trim($_POST['pass']),
FILTER_SANITIZE_STRING);

    if ($mysql->connect_error) {
        die("Помилка при підключенні до бази даних: " . $mysql-
>connect_error);
    }

    $result = $mysql->query("SELECT * FROM `users` WHERE `login` =
'$login' AND `password` = '$password'");

    if (!$result) {
        die("Помилка при виконанні запроса: " . $mysql->error);
    }

    $user = $result->fetch_assoc();

    if ($user === null) {
        echo "Такий користувач не знайдено";
        exit;
    }

    if (!empty($user['login'])) {
        setcookie('user', $user['login'], time() + 3600, "/");
    } else {
        echo "Ім'я користувача пусте.";
        exit;
    }

    setcookie('user', $user['login'], time() + 3600, "/");

    $mysql->close();

    header('Location: index.php');
    ?>

```

Лістинг А.4– Файл name.index.php

```

<?php
require_once 'db.php';

$result = $mysql->query("SELECT * FROM `books`");
?>
<!DOCTYPE html>
<html lang="UA">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>CHITANKA</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">

```

```

        <link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
        <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;7
00&display=swap" rel="stylesheet">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.7.1/css/all.min.css">
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css">
        <link rel="stylesheet" href="assets/css/main.css">
    </head>
    <body>
        <div class="wrapper">

            <header class="header">
                <div class="header-top py-2">
                    <div class="container-fluid">
                        <div class="row align-items-center">

                            <div class="col-6 col-sm-4">
                                <div class="header-top-phone d-flex
align-items-center h-100">
                                    <i class="fa-solid fa-
phone"></i>
                                    <a href="tel:+1234567890"
class="ms-2">123-456-7890</a>
                                </div>
                            </div>

                            <div class="col-12 col-sm-4 text-center
my-2 my-sm-0">
                                <a href="index.php" class="header-
logo h2 mb-0 fw-bold d-inline-block border border-success border-2
rounded-pill px-3 py-1 text-declaration-none">CHITANKA</a>
                            </div>

                            <div class="col-6 col-sm-4">
                                <div class="header-top-account d-flex
justify-content-end">
                                    <div class="btn-group me-2">
                                        <?php if
(!isset($_COOKIE['user'])): ?>
                                            <div class="dropdown">
                                                <button class="btn btn-md dropdown-
toggle" type="button" data-bs-toggle="dropdown" aria-
expanded="false">
                                                    Аккаунт
                                                </button>
                                                <ul class="dropdown-menu">
                                                    <li><a class="dropdown-item"
href="login.php">Увійти</a></li>

```

```

                                <li><a class="dropdown-item"
href="register.php">Зареєструватись</a></li>
                                </ul>
                                </div>
                                <?php else: ?>
                                <div>
                                    <span class="me-2"> <?php echo
htmlspecialchars($_COOKIE['user']); ?></span>
                                    <a href="logout.php" class="btn btn-
md btn-don">Вийти</a>
                                </div>
                                <?php endif; ?>
                                </div>
                                <div class="btn-group">
                                    <div class="dropdown">
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>

                    <div class="offcanvas offcanvas-end" tabindex="-1"
id="offcanvasCart" aria-labelledby="offcanvasCartLabel">
                        <div class="offcanvas-header">
                            <h5 class="offcanvas-title"
id="offcanvasCartLabel">Кошик</h5>
                            <button type="button" class="btn-close" data-
bs-dismiss="offcanvas" aria-label="Close"></button>
                        </div>
                        <div class="offcanvas-body">

                            <div class="table-responsive">
                                <table class="table offcanvasCart-table">
                                    <tbody id="cart-items">
                                        </tbody>
                                    </tbody>
                                <tfoot>
                                    <tr>
                                        <td colspan="4" class="text-end">Усього:</td>
                                        <td id="cart-total">$0</td>
                                    </tr>
                                </tfoot>
                            </table>
                        </div>
                        <div class="text-end">
                            <a href="checkout.php" class="btn btn-outline-
secondary">Оформити замовлення</a>
                        </div>
                    </div>
                </div>
            </div>

```

```

</header>

<div class="header-bottom sticky-top" id="header-nav" >
  <nav class="navbar navbar-expand-lg bg-dark" data-bs-
theme="dark">
    <div class="container-fluid">
      <button class="navbar-toggler" type="button"
data-bs-toggle="offcanvas"
      data-bs-target="#offcanvasNavbar" aria-
controls="offcanvasNavbar" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="offcanvas offcanvas-start"
id="offcanvasNavbar" tabindex="-1" aria-
labelledby="offcanvasNavbarLabel">

        <div class="offcanvas-header">
          <h5 class="offcanvas-title"
id="offcanvasNavbarLabel">Каталог</h5>
          <button type="button" class="btn-close" data-
bs-dismiss="offcanvas" aria-label="Close"></button>
        </div>

        <div class="offcanvas-body">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link active" aria-
current="page" href="index.php">Головна</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#about" data-
href="about">Про нас</a>
            </li>
            <li class="nav-item">
              <a class="nav-link"
href="category.php">Каталог</a>
            </li>
          </ul>
        </div>
      </div>

      <div>

        <button class="btn p-1" id="cart-open"
type="button" data-bs-toggle="offcanvas" data-bs-
target="#offcanvasCart" aria-controls="offcanvasCart">
          <i class="fa-solid fa-cart-shopping"></i>
          <span class="badge text-bg-success cart-badge
bg-success rounded-circle" id="cart-badge">0</span>
        </button>
      </div>
    </div>
  </div>

```

```

        </nav>
    </div>

    <main class="main">

        <div id="carousel" class="carousel slide">
            <div class="carousel-indicators">
                <button type="button" data-bs-target="#carousel"
data-bs-slide-to="0" class="active" aria-current="true" aria-
label="Slide 1"></button>
                <button type="button" data-bs-target="#carousel"
data-bs-slide-to="1" aria-label="Slide 2"></button>
                <button type="button" data-bs-target="#carousel"
data-bs-slide-to="2" aria-label="Slide 3"></button>
            </div>
            <div class="carousel-inner">
                <div class="carousel-item active">
                    
                </div>
                <div class="carousel-item">
                    
                </div>
                <div class="carousel-item">
                    
                </div>
            </div>
            <button class="carousel-control-prev" type="button"
data-bs-target="#carousel" data-bs-slide="prev">
                <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
                <span class="visually-hidden">Пред</span>
            </button>
            <button class="carousel-control-next" type="button"
data-bs-target="#carousel" data-bs-slide="next">
                <span class="carousel-control-next-icon" aria-
hidden="true"></span>
                <span class="visually-hidden">След</span>
            </button>
        </div>

        <section class="featured-products px-lg-3">
            <div class="container-fluid">
                <div class="row mb-5">
                    <div class="col-12">
                        <h2 class="section-title">
                            <span>Рекомендовані товари</span>
                        </h2>
                    </div>
                </div>
            </div>
        </div>
    </main>

```

```

        <div class="row">
            <?php
                $featured_query = $mysql->query("SELECT * FROM
`books` ORDER BY RAND() LIMIT 3");
                while ($book = $featured_query->fetch_assoc()):
                    ?>
                    <div class="col-lg-4 col-sm-6 mb-4 product" data-
category="<?= htmlspecialchars($book['category']) ?>">
                        <div class="product-card h-100 d-flex flex-
column">

                            <div class="product-thumb thumb-
standardize">
                                <a href="product.php?id=<?= $book['id']
?>">
                                    " class="img-contain">
                                        </a>
                                    </div>

                                    <div class="product-details flex-grow-1">
                                        <h4>
                                            <a href="product.php?id=<?=
$book['id'] ?>"><?= htmlspecialchars($book['title']) ?></a>
                                        </h4>
                                    </div>

                                    <div class="product-bottom-details d-flex
justify-content-between align-items-center mb-3">
                                        <div class="product-price ms-3 text-
danger fw-bold">
                                            <?=
htmlspecialchars($book['price']) ?> ₺
                                        </div>
                                        <div class="product-links me-3">
                                            <button class="btn btn-outline-
secondary add-to-cart"
                                                data-id="<?= $book['id']
?>"
                                                data-title="<?=
htmlspecialchars($book['title']) ?>"
                                                data-price="<?=
$book['price'] ?>">
                                                <i class="fas fa-shopping-
cart"></i>
                                            </button>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        <?php endwhile; ?>

```

```

        </div>
    </div>
</section>

</section>
<section class="about-us" id="about">
    <div class="container fluid">
        <div class="row mb-5">
            <div class="col-12">
                <h2 class="section-title">
                    <span>Про нас</span>
                </h2>
            </div>
        </div>
        <div class="row">
            <div class="col-12">
                <p> Сайт книжного магазина "ЧИТАНКА" </p>
                <p> Опис.... </p>

            </div>
        </div>
    </div>
</section>

</main>
<footer class="footer py-4">
    <div class="container-fluid">
        <div class="row">

            <div class="col-md-4 col-6 ps-md-5 ps-3 mb-3">
                <h4>Інформація</h4>
                <ul class="list-unstyled">
                    <li><a href="index.php">Головна</a></li>
                    <li><a href="category.php">Каталог</a></li>
                </ul>
            </div>

            <div class="col-md-4 col-6 mb-3">
                <h4>Контакти</h4>
                <ul class="list-unstyled">
                    <li><a href="тел:1234567890">123-456-
7890</a></li>
                    <li><a href="тел:0987654321">098-765-
4321</a></li>
                </ul>
            </div>

            <div class="col-md-4 col-12">
                <h4>Слідкуй за новинами</h4>
                <ul class="footer-icons ">
                    <li><a href="#"><i class="fa-brands fa-
youtube"></i></a></li>

```

```

        <li><a href="#"><i class="fa-brands fa-
facebook-f"></i></a></li>
    </ul>
</div>
</div>
</div>
</footer>
</div>

```

```

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.
bundle.min.js"></script>
    <script src="assets/js/main.js"></script>
</body>
</html>

```

ДОДАТОК В

СЛАЙДИ ПРЕЗЕНТАЦІЇ

Розробка комп'ютерної системи інтернет-магазину книжок

дипломна робота бакалавра

Виконав:
Студент КНТ-513СП

Тіщенко Вадим Олексійович

Керівник:
Доцент, к.н.т.

Скрупський Степан Юрійович

Рисунок В.1 – Слайд 1

2/14

| Мета та завдання роботи

Предмет розробки - комп'ютерна система інтернет-магазину книжкової продукції «CHITANKA».

Мета роботи - підвищення ефективності онлайн-продажу книжок шляхом скорочення часу оформлення замовлень та оптимізації серверних ресурсів.

Новизна роботи - Використанням реактивного JavaScript-інтерфейсу з асинхронними взаємодіями з сервером через Fetch API без використання готових і «важких» CMS

Практична цінність роботи полягає у створенні процесдатного програмного продукту, який готовий до впровадження та модифікацій у сфері книготоргівлі.

Рисунок В.2 – Слайд 2

| Актуальність теми

60%

замовлень в e-commerce вже здійснюються через смартфони (Mobile-first)

Сучасний ринок книготоргівлі вимагає:

- ☐ Максимальної швидкості взаємодії.
- ☐ Адаптивності під будь-які пристрої.
- ☐ Безпеки персональних даних.

Рисунок В.3 – Слайд 3

| Аналіз існуючих аналогів

Параметр порівняння	Yakaboo	Книгарня «Є»	Клуб Сімейного Дозвілля
Інтерфейс	Перевантажений рекламою	Лаконічний, зручний	Застарілий дизайн
Технологічність	Висока (Власне ПЗ)	Середня	Низька
Мобільний додаток	Наявний	Відсутній	Відсутній
UX-фокус	Асортимент	Сервіс та бренд	Лояльність клубу

Рисунок В.4 – Слайд 4

| Архітектура системи

Обрано трирівневу клієнт-серверну модель:

- ❑ Рівень представлення: HTML5, CSS3, Bootstrap 5, Vanilla JS.
- ❑ Рівень логіки: PHP 8.x (обробка Fetch-запитів).
- ❑ Рівень даних: СУБД MySQL (реляційна структура).

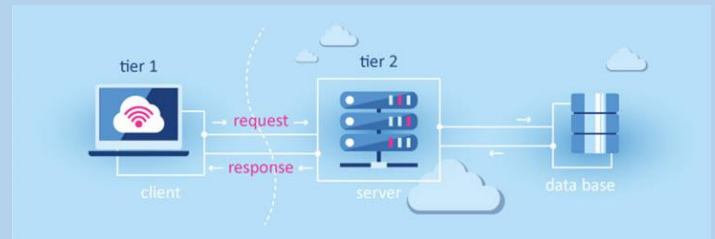


Рисунок В.5 – Слайд 5

| Проєктування бази даних

ER-діаграма системи включає 4 основні сутності:

- ❑ users: облікові дані та сесії.
- ❑ books: каталог та ціни.
- ❑ orders: "шапка" замовлення.
- ❑ order_items: деталізація позицій.

Цілісність даних забезпечена каскадними зовнішніми ключами.

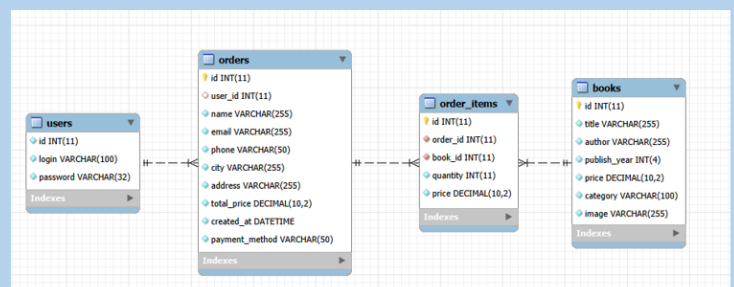


Рисунок В.6 – Слайд 6

Алгоритм оформлення замовлення

Використання асинхронної взаємодії через Fetch API:

- ❑ Клієнтська валідація (Bootstrap).
- ❑ Формування JSON-пакета.
- ❑ PHP-обробник save_order.php.
- ❑ Транзакційний запис у БД.
- ❑ Очищення LocalStorage після успіху.

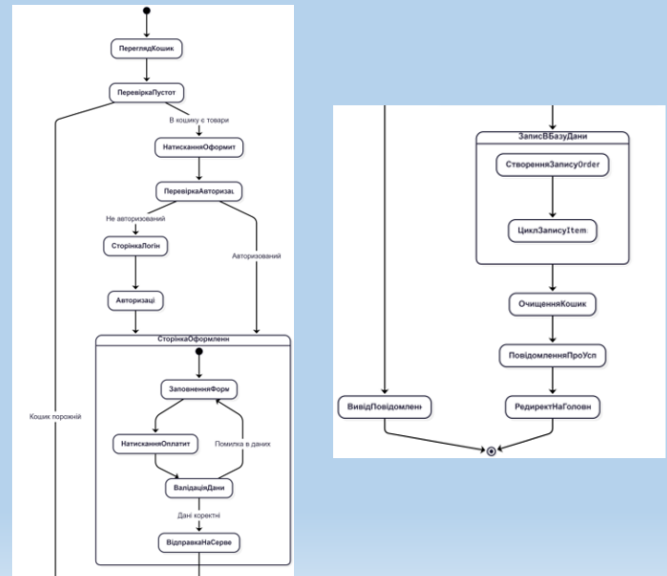


Рисунок В.7 – Слайд 7

Діаграма прецедентів системи

Актори та сценарії:

- ❑ Гість: перегляд каталогу, фільтрація за категоріями, додавання в кошик, реєстрація.
- ❑ Авторизований користувач: ініціювання та підтвердження замовлення.
- ❑ Система: перевірка залишків, розрахунок суми.

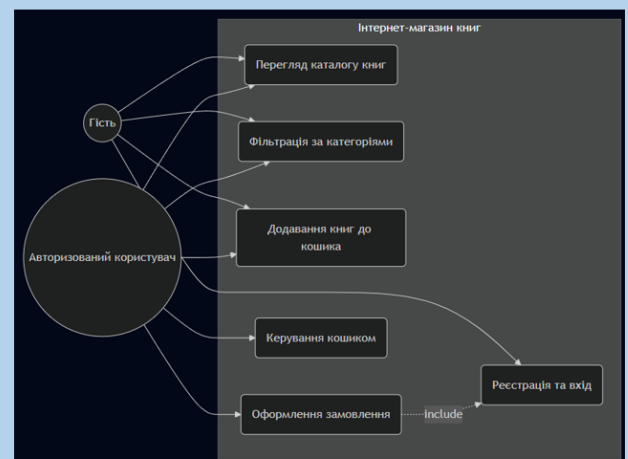


Рисунок В.8 – Слайд 8

| Технологічний стек розробки



Рисунок В.9 – Слайд 9

| Програмна реалізація

Ключові програмні модулі:

- ☐ auth.php: авторизація та Cookie.
- ☐ check.php: реєстрація та валідація.
- ☐ save_order.php: транзакційне збереження.
- ☐ main.js: рендеринг кошика та фільтри.

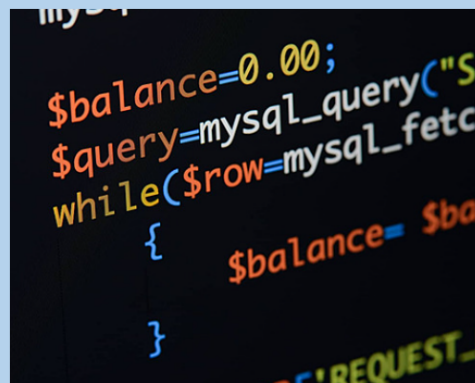
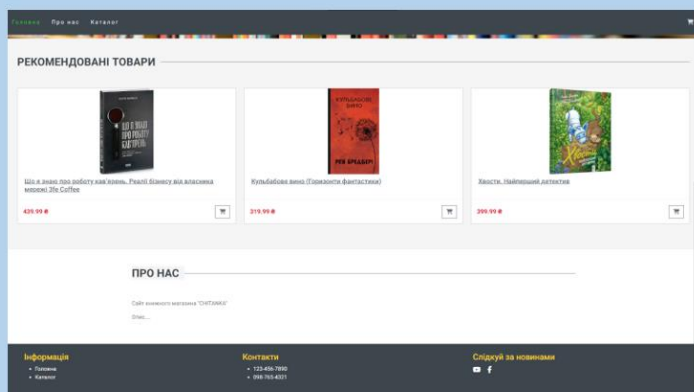


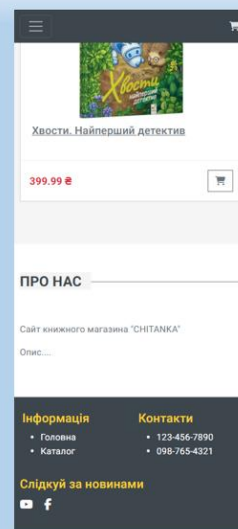
Рисунок В.10 – Слайд 10

11/14

Тестування адаптивності інтерфейсу



Десктопна версія



Мобільна версія (Pixel 7)

Рисунок В.11 – Слайд 11

12/14

Аудит цілосності бази даних

Результати перевірки в phpMyAdmin:

- ❑ Успішна прив'язка `user_id` до замовлення.
- ❑ Каскадне створення записів у `order_items`.
- ❑ Автоматична фіксація `created at`.

Extra options									
+ - +									
			id	user_id	name	email	phone	city	address
☐	Известия	Консультант	Удальцов	1	0	Bajane	viskupvatini@gmail.com	Запорожье	Запорожье 6
☐	Известия	Консультант	Удальцов	2	0	TK	viskupvatini@gmail.com	Запорожье	Запорожье 6
☐	Известия	Консультант	Удальцов	3	NULI	nyukle	viskupvatini@gmail.com	Запорожье	Запорожье 2
☐	Известия	Консультант	Удальцов	4	NULI	krasib	viskupvatini@gmail.com	Запорожье	Запорожье 6
☐	Известия	Консультант	Удальцов	5	8	Kuq	viskupvatini@gmail.com	Запорожье	Запорожье 5
☐	Известия	Консультант	Удальцов	6	9	jam	evse@gmail.com	Запорожье	Запорожье 6
☐	Известия	Консультант	Удальцов	7	9	snura	zozoz@gmail.com	Запорожье	Запорожье 5
☐	Известия	Консультант	Удальцов	8	10	doridon	wgczpzyvpcv@gmail.com	Зап	Зап
☐	Известия	Консультант	Удальцов	9	8	perseiva	wgczpzyvpcv@gmail.com	Зап	Зап
389 98 2026-05-12 15:34:29 Оплата кармано									
+ - +									
			id	order_id	book_id	quantity	price		
☐	Известия	Консультант	Удальцов	1	1	4	3 399.99		
☐	Известия	Консультант	Удальцов	2	2	5	1 399.99		
☐	Известия	Консультант	Удальцов	3	2	4	1 399.99		
☐	Известия	Консультант	Удальцов	4	2	6	1 430.99		
☐	Известия	Консультант	Удальцов	5	3	6	1 430.99		
☐	Известия	Консультант	Удальцов	6	3	4	2 399.99		
☐	Известия	Консультант	Удальцов	7	3	5	3 399.99		
☐	Известия	Консультант	Удальцов	8	5	1	1 319.99		
☐	Известия	Консультант	Удальцов	9	5	2	1 319.99		
☐	Известия	Консультант	Удальцов	10	5	3	1 440.99		
☐	Известия	Консультант	Удальцов	11	6	1	1 319.99		
☐	Известия	Консультант	Удальцов	12	7	2	1 529.99		
☐	Известия	Консультант	Удальцов	13	7	5	2 399.99		
☐	Известия	Консультант	Удальцов	14	14	7	2 319.99		
☐	Известия	Консультант	Удальцов	15	7	2	1 529.99		
☐	Известия	Консультант	Удальцов	16	8	1	4 319.99		
☐	Известия	Консультант	Удальцов	17	8	4	2 399.99		
☐	Известия	Консультант	Удальцов	18	9	6	1 430.99		
☐	Известия	Консультант	Удальцов	19	9	5	1 399.99		

Рисунок В.12 – Слайд 12

| Результати функціонального тестування

№	Об'єкт тестування	Дія користувача	Результат
1	Модуль кошика	Натискання кнопки візка	Товар додано до LocalStorage
2	Фільтрація	Вибір категорії	Миттєве приховування d-none
3	Автентифікація	Вхід в акаунт	Встановлення Cookie "user"
4	Оформлення	Клік "Оплатити"	Запис у БД через Fetch API

Рисунок В.13 – Слайд 13

| Висновки

- ✓ Розроблено та впроваджено функціональний вебдодаток інтернет-магазину «CHITANKA».
- ✓ Використання 3-рівневої архітектури забезпечило стабільність та масштабованість системи.
- ✓ Реалізовано "безшовний" UX завдяки технологіям асинхронної передачі даних.
- ✓ Система успішно пройшла всі етапи тестування та готова до практичної експлуатації.

Рисунок В.14 – Слайд 14