

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО
«ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
“МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ”»

ФАКУЛЬТЕТ КОМП'ЮТЕРНО-ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**«Проектування та розробка інформаційної системи
продажу та обміну книг у форматі мобільного
вебзастосунку».**

Шифр групи: ІК-9-21-Б1ПЗ(4.0д)

Лигирда Олександра Василівна

Спеціальність:

Інженерія програмного забезпечення

Робота на здобуття освітньо-кваліфікаційного рівня бакалавр

Науковий керівник

(підпис)

Яременко Д. М.

Допущено до захисту ДЕК

Завідувач кафедри

(підпис)

професор, док. екон.
наук
Кавун С.В.

Київ 2026

АНОТАЦІЯ

Кваліфікаційна робота присвячена проєктуванню та розробці інформаційної системи продажу та обміну книг у форматі мобільного вебзастосунку «BookShelf».

У роботі розглянуто процес створення вебзастосунку, який дає змогу користувачам реєструватися, переглядати каталог книг, створювати власні оголошення, фільтрувати їх за містом і типом, а також зв'язуватися з продавцем через email. Під час розробки було використано HTML, CSS, Bootstrap, JavaScript, Node.js, Express.js та MySQL.

Результатом роботи є функціональний мобільний вебзастосунок, який може бути використаний для організації продажу та обміну книг між користувачами.

Ключові слова: інформаційна система, мобільний вебзастосунок, BookShelf, продаж книг, обмін книг, оголошення, адаптивний інтерфейс, Node.js, Express.js, MySQL.

ABSTRACT

This qualifying work is devoted to the design and development of an information system for the sale and exchange of books in the form of a mobile web application called 'BookShelf'.

The thesis examines the process of creating a web application that allows users to register, browse the book catalogue, create their own listings, filter them by city and type, and contact sellers through email. HTML, CSS, Bootstrap, JavaScript, Node.js, Express.js and MySQL were used during development.

The result of this work is a functional mobile web application that can be used to organise the sale and exchange of books between users.

Keywords: information system, mobile web application, BookShelf, book sales, book exchange, listings, responsive interface, Node.js, Express.js, MySQL.

ЗМІСТ

ВСТУП.....	5
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОДАЖУ ТА ОБМІНУ КНИГ.....	7
1.1 Аналіз предметної області продажу та обміну книг в онлайн-середовищі.....	7
1.2 Особливості мобільних вебзастосунків для взаємодії користувачів.....	9
1.3 Огляд існуючих рішень для продажу, обміну та пошуку книг.....	11
1.4 Обґрунтування доцільності розробки інформаційної системи BookShelf.....	13
Висновки до розділу 1.....	14
 РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ВЕБЗАСТОСУНКУ «BookShelf»	16
2.1. Функціональні і нефункціональні вимоги до застосунку.....	16
2.2. Сценарії варіантів використання застосунку.....	19
2.3. Проєктування архітектури застосунку.....	21
2.4. Проєктування бази даних та API для взаємодії клієнтської і серверної частин.....	25
Висновки до розділу 2.....	28
 РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОБІЛЬНОГО ВЕБЗАСТОСУНКУ «BookShelf».....	30
3.1. Вибір технологій для розробки застосунку.....	30
3.2. Реалізація архітектурного каркасу застосунку.....	31
3.3. Реалізація клієнтської частини мобільного вебзастосунку.....	33
3.4. Реалізація серверної частини, авторизації та роботи з оголошеннями.....	36
Висновки до розділу 3.....	41

РОЗДІЛ 4. ТЕСТУВАННЯ МОБІЛЬНОГО ВЕБЗАСТОСУНКУ.....	42
4.1. Методика і проведення тестування застосунку.....	42
4.2. Результати тестування основних функціональних можливостей.....	44
4.3 Аналіз виявлених недоліків та напрями подальшого розвитку застосунку.....	53
Висновки до розділу 4.....	56
ЗАГАЛЬНІ ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	59
ДОДАТКИ.....	64

ВСТУП

Актуальність теми

У сучасних умовах вебзастосунки стали одним із найпоширеніших способів організації взаємодії між користувачами в різних сферах: торгівлі, освіті, комунікації, обміні товарами та послугами. Значна частина користувачів надає перевагу саме мобільному формату роботи з вебресурсами, оскільки смартфон дозволяє швидко отримувати доступ до потрібної інформації, переглядати оголошення, виконувати пошук і зв'язуватися з іншими користувачами.

Окремої уваги потребує сфера продажу та обміну книг. Попри розвиток електронних видань, друковані книги залишаються актуальними для навчання, дозвілля, колекціонування та особистого користування. Водночас у багатьох користувачів є книги, які вони вже прочитали і готові продати або обміняти. Інші користувачі, навпаки, шукають потрібні книги за доступнішою ціною або хочуть знайти видання у своєму місті.

Проблема полягає в тому, що не всі наявні онлайн-платформи орієнтовані саме на простий обмін і продаж книг між звичайними користувачами. Часто такі сервіси мають надмірну кількість функцій, складні форми створення оголошень або не враховують потребу в швидкому пошуку за містом, типом оголошення чи назвою книги. Тому виникає потреба у створенні зручної інформаційної системи, яка буде орієнтована саме на базові дії користувача: зареєструватися, переглянути каталог, знайти потрібну книгу, створити власне оголошення та зв'язатися з продавцем.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю розробки адаптивного мобільного вебзастосунку, який поєднує простий інтерфейс, базові функції керування оголошеннями та можливість швидкої взаємодії між користувачами. Саме таким рішенням є вебзастосунок «BookShelf», призначений для продажу та обміну книг.

Мета роботи

Метою кваліфікаційної роботи є проєктування та розробка інформаційної

системи продажу та обміну книг у форматі мобільного вебзастосунку «BookShelf».

Завдання роботи

Для досягнення поставленої мети потрібно виконати такі завдання:

1. Проаналізувати предметну область продажу та обміну книг.
2. Визначити функціональні та нефункціональні вимоги до застосунку.
3. Спроекувати структуру мобільного вебзастосунку.
4. Спроекувати базу даних для зберігання користувачів та оголошень.
5. Реалізувати клієнтську частину застосунку.
6. Реалізувати серверну частину та API.
7. Додати можливість реєстрації, входу, створення, редагування та видалення оголошень.
8. Реалізувати пошук і фільтрацію оголошень.
9. Провести тестування основних функцій застосунку.

Об'єкт дослідження: Об'єктом дослідження є процес продажу та обміну книг між користувачами за допомогою вебтехнологій.

Предмет дослідження: Предметом дослідження є мобільний вебзастосунок для створення, перегляду та керування оголошеннями про книги.

Практичне значення роботи

Практичне значення роботи полягає у створенні вебзастосунку «BookShelf», який дозволяє користувачам розміщувати оголошення про продаж або обмін книг, переглядати доступні пропозиції та зв'язуватися з продавцями.

Структура роботи

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОДАЖУ ТА ОБМІНУ КНИГ

1.1 Аналіз предметної області продажу та обміну книг в онлайн-середовищі

Продаж та обмін книг є частиною ширшої сфери електронної комерції та цифрової взаємодії між користувачами. У традиційному розумінні купівля книг відбувається через книгарні, видавництва, інтернет-магазини або маркетплейси. Однак поряд із цим активно існує інший напрям — продаж і обмін уживаних книг між звичайними користувачами. Такий формат є зручним для людей, які вже прочитали книгу і хочуть передати її іншому читачеві, продати за нижчою ціною або обміняти на інше видання.

Актуальність такої предметної області пов'язана з тим, що книжковий ринок України продовжує розвиватися навіть в умовах складної економічної та соціальної ситуації. За даними Українського інституту книги, у 2024 році дохід видавців збільшився на 31% порівняно з 2023 роком, загальний наклад зріс на 21%, а в Україні було видано 15 601 книжкову назву із загальним накладом понад 33 млн примірників. Також зазначено, що в Україні працює близько 800 фізичних книгарень і приблизно 80 онлайн-магазинів книг. Це свідчить про те, що попит на книжкову продукцію залишається стабільним, а цифрові канали продажу мають важливе значення для книжкового ринку. [4]

Онлайн-середовище дає змогу значно спростити процес пошуку та передачі книг між користувачами. Якщо раніше для продажу або обміну книги людина мала звертатися до знайомих, бібліотек, букіністичних магазинів або локальних груп, то зараз такі дії можна виконувати через вебсайти, мобільні застосунки, соціальні мережі, месенджери та платформи оголошень. Це дозволяє швидше знайти потенційного покупця або людину для обміну, а також розширює коло користувачів за межі особистого оточення.

Важливим чинником розвитку таких сервісів є високий рівень

користування інтернетом і мобільним зв'язком. За даними DataReportal, на кінець 2025 року в Україні було 35,3 млн користувачів інтернету, а рівень проникнення інтернету становив 89,6% населення. Також в Україні було зафіксовано 55 млн мобільних підключень, що еквівалентно 140% населення, оскільки одна людина може використовувати кілька SIM-карт або підключень. [11] Це підтверджує доцільність створення саме мобільного вебзастосунку, оскільки значна частина користувачів має доступ до інтернету через смартфон.

Предметна область продажу та обміну книг має свої особливості. На відміну від звичайної торгівлі новими товарами, тут важливо враховувати стан книги, її фізичний вигляд, місто розташування власника, можливість обміну, ціну, опис та додаткові примітки продавця. Для користувача важливо не лише побачити назву книги, а й зрозуміти, чи вона нова, у доброму стані, вживана, чи підходить для обміну, а також як можна зв'язатися з власником.

Основними учасниками такої системи є:

- користувач, який хоче продати або обміняти книгу;
- користувач, який шукає книгу;
- система, яка зберігає оголошення, дані користувачів і забезпечує взаємодію між ними.

Продаж і обмін книг в онлайн-середовищі також має певні проблеми. Перша проблема — розпорошеність оголошень між різними платформами: частина користувачів розміщує книги на загальних сервісах оголошень, частина — у соціальних мережах або месенджерах, частина — у локальних групах. Друга проблема — відсутність спеціалізованої структури саме для книг. Наприклад, на загальному маркетплейсі книга є лише одним із багатьох типів товарів, тому сервіс не завжди орієнтований на специфіку книжкових оголошень. Третя проблема — складність швидкого пошуку за параметрами, які важливі саме для читача: місто, стан книги, тип пропозиції, назва, можливість обміну.

Отже, предметна область продажу та обміну книг потребує

інформаційної системи, яка враховує специфіку книжкових оголошень: стан примірника, місто, тип пропозиції, опис, ціну або умови обміну та спосіб зв'язку з власником. Саме ці параметри мають бути покладені в основу подальшого проектування BookShelf.

1.2 Особливості мобільних вебзастосунків для взаємодії користувачів

Мобільний вебзастосунок — це вебресурс, який відкривається через браузер на смартфоні або планшеті та адаптується до розміру екрана користувача. На відміну від нативного мобільного застосунку, мобільний вебзастосунок не потребує встановлення, що полегшує початок використання. Користувачам достатньо відкрити застосунок в браузері. Особливо це зручно для тимчасового використання. [5]

Такий формат є доцільним, оскільки він простіший у розробці та підтримці, ніж окремі нативні застосунки для Android та iOS. Один вебзастосунок може працювати на різних пристроях, якщо його інтерфейс правильно адаптований. Google у своїй документації зазначає, що responsive web design використовує один HTML-код і одну URL-адресу для різних пристроїв, але змінює відображення залежно від розміру екрана. Також Google рекомендує адаптивний дизайн як найпростіший для реалізації та підтримки підхід. [28]

Однією з головних особливостей мобільних вебзастосунків є обмежений простір екрана. На смартфоні неможливо зручно розмістити велику кількість колонок, складні таблиці або довгі меню. Тому інтерфейс має бути простим, структурованим і зрозумілим. Для мобільних пристроїв часто використовується підхід mobile-first — це методологія, у якій дизайн спочатку створюється для мобільних пристроїв, а вже потім масштабують під планшети й десктопи. [27] У документації MDN зазначено, що типовим підходом є створення простої одноколонкової структури для мобільних пристроїв, після чого для ширших екранів можна додавати складнішу багатокolonкову структуру. [35]

Ще однією важливою особливістю є використання адаптивної верстки. Вона дозволяє сторінкам коректно відображатися на різних екранах. Для цього застосовуються гнучкі сітки, відносні одиниці вимірювання, адаптивні зображення, медіазапити CSS та компоненти, які змінюють своє розташування залежно від ширини екрана. MDN зазначає, що медіазапити допомагають реалізувати responsive web design, однак адаптивність також може будуватися на гнучких сітках, відносних одиницях та мінімальних і максимальних значеннях розмірів. [35]

Для системи продажу та обміну книг мобільний формат особливо важливий, оскільки користувач може швидко переглянути оголошення, створити власну пропозицію, перевірити інформацію про книгу або зв'язатися з продавцем без комп'ютера. Наприклад, людина може сфотографувати книгу на телефон, заповнити коротку форму та одразу додати оголошення. Інший користувач може знайти книгу за назвою або містом і швидко написати продавцю.

Також для мобільних вебзастосунків важлива швидкість роботи. Google індексує сайти за mobile-first принципом, тому швидкість завантаження та якість мобільної версії стали ключовими факторами ранжування. Повільні сторінки, важкі зображення, складна навігація – усе це знижує позиції в пошуковій видачі та погіршує поведінкові показники. Mobile First допомагає оптимізувати структуру, контент і навігаційні сценарії так, щоб сторінки відкривались швидко навіть на слабких пристроях і нестабільному мобільному інтернеті. [27] W3C у рекомендаціях щодо мобільних вебзастосунків наголошує на важливості оптимізації часу відповіді, гнучкості дизайну, ефективного використання мережі та врахування особливостей мобільних пристроїв. [29]

Отже, мобільний вебформат є доцільним для BookShelf, оскільки дає змогу працювати із каталогом, формами та профілем без встановлення окремого застосунку. Це відповідає завданню створити доступний інструмент

для швидкої взаємодії між користувачами.

1.3 Огляд існуючих рішень для продажу, обміну та пошуку книг

Перед розробкою інформаційної системи BookShelf доцільно розглянути наявні рішення, які частково виконують схожі функції. До таких рішень можна віднести загальні сервіси оголошень, книжкові онлайн-магазини, міжнародні платформи з продажу вживаних книг, сервіси буккросингу та книжкові соціальні платформи.

Одним із найпоширеніших прикладів загального сервісу оголошень в Україні є OLX. На платформі є окрема категорія «Книги / журнали», де користувачі можуть розміщувати оголошення про продаж книг, журналів, коміксів і манги. На сторінці категорії доступні фільтри за фото, доставкою, ціною, станом та типом продавця, а сама платформа показує понад 1000 оголошень у цій категорії.[2] Перевагою OLX є велика аудиторія, знайомий для користувачів інтерфейс і можливість продавати різні товари. Однак недоліком є те, що OLX не є спеціалізованим книжковим сервісом. Книги там розміщуються поряд з іншими товарами, а сама система не зосереджена саме на обміні книгами між читачами.

Іншим типом рішень є книжкові онлайн-магазини, наприклад Yakaboo. Це національна книжкова платформа України, яка позиціонує себе як найбільший книжковий онлайн-магазин в Україні. Проте Yakaboo не вирішує задачу продажу власних уживаних книг звичайними користувачами й не орієнтований на прямий обмін між читачами.

До міжнародних спеціалізованих сервісів можна віднести AbeBooks. Платформа спеціалізується на продажу нових, уживаних, рідкісних і колекційних книг. На сайті зазначено, що там представлені вживані книги, рідкісні видання, книги від незалежних продавців з усього світу, а також інші предмети колекціонування.[6] Такий сервіс демонструє, що ринок уживаних і рідкісних книг може бути організований як окрема цифрова платформа.

Окремим напрямом є BookCrossing. Його ідея полягає не у продажу, а у

вільному передаванні книг іншим людям. На офіційному сайті BookCrossing зазначено, що сервіс дозволяє «відпускати» книги у вільний обіг або передавати їх іншому учаснику, а потім відстежувати шлях книги через записи користувачів.

В Україні також існують локальні ініціативи буккросингу. Наприклад, у матеріалі Chytomo зазначалося про Telegram-сервіс безкоштовного обміну друкованими книжками між українськими читачами, який складався з каналу з каталогом книжок і бота для приймання книг із фото. [3] Крім того, бібліотеки також можуть організовувати полиці буккросингу. Наприклад, Чернігівська обласна універсальна наукова бібліотека описує буккросинг як можливість вільного обміну власними прочитаними книгами, яким читачі хочуть продовжити життя.[1] Такі приклади показують, що інтерес до обміну книгами існує, але часто він реалізований не як повноцінна інформаційна система, а як локальна ініціатива, група, канал або фізична полиця.

Ще одним типом книжкових сервісів є Goodreads. Це міжнародна платформа для пошуку книг, ведення читацьких списків, отримання рекомендацій і перегляду того, що читають друзі. На сайті Goodreads зазначено, що сервіс допомагає користувачам вирішити, що читати далі, шукати й переглядати книги за жанрами, а також отримувати рекомендації на основі прочитаного. Проте Goodreads не є сервісом для продажу або обміну фізичними книгами між користувачами. Його головна функція — читацька соціальна взаємодія та рекомендації.

Проведений огляд показує, що на ринку існують різні типи рішень: загальні маркетплейси, онлайн-книгарні, міжнародні сервіси вживаних книг, буккросингові ініціативи та читацькі соціальні мережі. Проте кожне з них вирішує лише частину задачі. Загальні маркетплейси не мають книжкової спеціалізації, онлайн-магазини не дозволяють звичайним користувачам продавати власні книги, буккросинг не передбачає продажу, а соціальні книжкові платформи не орієнтовані на фізичну передачу книг між користувачами.

1.4 Обґрунтування доцільності розробки інформаційної системи BookShelf

Доцільність розробки BookShelf визначається тим, що система зосереджена на конкретному користувацькому сценарії: розміщенні, пошуку та перегляді книжкових оголошень. На відміну від великих маркетплейсів або класичних інтернет-магазинів, застосунок не охоплює повний цикл електронної комерції з кошиком, онлайн-оплатою та доставкою, а реалізує функції, необхідні саме для первинної взаємодії між власником книги й зацікавленим користувачем.

Доцільність розробки BookShelf можна пояснити кількома причинами.

По-перше, система має чітку предметну спеціалізацію. Структура оголошення передбачає поля, важливі саме для книги: назву, опис, стан примірника, тип оголошення, місто, фото, ціну та нотатку продавця. Завдяки цьому користувач отримує не загальну товарну картку, а запис, адаптований до книжкового контексту.

По-друге, система підтримує два типи взаємодії: продаж і обмін. Це важливо, оскільки не всі користувачі хочуть саме продавати книги. Частина читачів готова обмінюватися прочитаними виданнями, особливо якщо йдеться про художню літературу, навчальні матеріали, комікси, мангу або книги, які вже не використовуються.

По-третє, BookShelf враховує локальний контекст. Для користувача важливо знати місто, у якому знаходиться книга. Це дозволяє швидше домовитися про зустріч, самовивіз або зручний спосіб передачі. У великих загальних платформах така можливість також може існувати, але вона не завжди поєднана зі спеціалізованими книжковими параметрами.

По-четверте, формат мобільного вебзастосунку дозволяє користувачам працювати із системою через браузер на смартфоні. Це зменшує бар'єр входу для користувача та спрощує підтримку проєкту, оскільки не потрібно створювати окремі нативні застосунки для різних платформ.

По-п'яте, обсяг функціональності відповідає межам дипломного MVP. До реалізації включено реєстрацію, вхід, каталог оголошень, пошук, фільтрацію,

створення, редагування й видалення власних оголошень, профіль користувача та зв'язок із продавцем через email.

Важливим є й те, що в системі передбачено базові механізми захисту даних. Зокрема, паролі користувачів зберігаються у хешованому вигляді, користувач може редагувати та видаляти лише власні оголошення, а перед створенням оголошення перевіряється існування користувача. Це важливо для системи, у якій користувачі створюють власні записи та взаємодіють з особистими даними.

Подальший розвиток BookShelf може передбачати адміністративну панель, статуси оголошень, розширені фільтри, внутрішній чат і вдосконалену роботу із зображеннями. Однак ці функції не є обов'язковими для базової версії, оскільки основна мета MVP полягає в перевірці ключового сценарію продажу та обміну книг.

Висновки до розділу 1

У першому розділі проаналізовано предметну область продажу та обміну книг в онлайн-середовищі. Визначено, що користувачам потрібні інструменти для розміщення власних книг, пошуку пропозицій за важливими параметрами та швидкого зв'язку з власником оголошення.

Особливістю цієї предметної області є те, що для книжкового оголошення важливими є не лише назва та ціна, а й стан примірника, місто, тип пропозиції, опис, можливість обміну та контактні дані продавця.

Також розглянуто особливості мобільних вебзастосунків. Для BookShelf такий формат є виправданим, оскільки користувач може переглядати каталог, заповнювати форми й керувати своїми оголошеннями без встановлення окремого програмного забезпечення.

Під час огляду існуючих рішень було проаналізовано загальні сервіси оголошень, книжкові онлайн-магазини, міжнародні сервіси продажу вживаних книг, буккросингові ініціативи та читацькі платформи. Аналіз показав, що ці рішення частково виконують схожі функції, але не поєднують у собі простий

продаж, обмін, книжкову спеціалізацію, локальний пошук і мобільний вебінтерфейс.

Проведений аналіз дав підстави для подальшого проектування BookShelf як інформаційної системи з каталогом оголошень, профілем користувача, фільтрацією, CRUD-операціями та email-зв'язком між учасниками.

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ВЕБЗАСТОСУНКУ «BookShelf»

2.1. Функціональні і нефункціональні вимоги до застосунку

Проектування мобільного вебзастосунку «BookShelf» доцільно розпочати з визначення вимог до системи. Вимоги описують, які функції має виконувати програмний продукт, які обмеження потрібно врахувати під час розробки та яким очікуванням користувачів повинна відповідати система. У стандарті ISO/IEC/IEEE 29148 зазначено, що вимоги до систем і програмних продуктів мають формуватися протягом життєвого циклу розробки та використовуватися як основа для подальшого проєктування, реалізації й перевірки програмного забезпечення.[21]

Для інформаційної системи «BookShelf» вимоги поділено на функціональні та нефункціональні. Функціональні вимоги описують, що саме має робити система, тобто її основні можливості. IBM визначає функціональні вимоги як вимоги, що стосуються функціональності нового продукту, системи або застосунку та мають бути вимірюваними для перевірки відповідності очікуванням користувачів.[24]

Нефункціональні вимоги визначають не окремі функції, а якісні характеристики системи: зручність, продуктивність, безпеку, надійність, підтримуваність та адаптивність. Модель якості ISO/IEC 25010 виділяє такі характеристики як функціональна придатність, продуктивність, сумісність, зручність використання, надійність, безпека, супроводжуваність і переносимість.[23]

Основною метою застосунку «BookShelf» є надання користувачам можливості створювати, переглядати, шукати та керувати оголошеннями про продаж або обмін книг. Система не проєктується як повноцінний інтернет-магазин із кошиком, онлайн-оплатою та доставкою. Основний акцент робиться на створенні каталогу книжкових оголошень і забезпеченні зв'язку між користувачами.

№	Функціональна вимога	Опис
ФВ-1	Реєстрація користувача	Система повинна надавати можливість новому користувачу створити обліковий запис, вказавши ім'я, email, пароль і місто.
ФВ-2	Авторизація користувача	Система повинна дозволяти зареєстрованому користувачу виконати вхід за email і паролем.
ФВ-3	Перегляд каталогу книг	Користувач повинен мати можливість переглядати список доступних оголошень про книги.
ФВ-4	Перегляд детальної інформації про книгу	Система повинна надавати сторінку або блок із детальним описом книги, її станом, містом, типом оголошення та інформацією про продавця.
ФВ-5	Пошук книг за назвою	Користувач повинен мати можливість знаходити книги за введеною назвою або частиною назви.
ФВ-6	Фільтрація оголошень	Система повинна підтримувати фільтрацію оголошень за містом і типом пропозиції: продаж або обмін.
ФВ-7	Створення оголошення	Авторизований користувач повинен мати можливість створити нове оголошення про книгу.
ФВ-8	Редагування власного оголошення	Користувач повинен мати можливість змінювати лише ті оголошення, які були створені ним.
ФВ-9	Видалення власного оголошення	Користувач повинен мати можливість видаляти тільки власні оголошення.
ФВ-10	Особистий профіль	Система повинна мати сторінку профілю користувача з основною інформацією та переліком його оголошень.
ФВ-11	Зв'язок із продавцем	Система повинна надавати можливість зв'язатися з власником оголошення через email.
ФВ-12	Вихід із системи	Авторизований користувач повинен мати можливість завершити сеанс роботи із системою.
ФВ-13	Повідомлення про помилки	Система повинна повідомляти користувача про неправильне заповнення форм, помилки авторизації або відсутність потрібних даних.

Таблиця 2.1 — Функціональні вимоги до застосунку

Особливу увагу під час проектування потрібно приділити авторизації та контролю доступу. Користувач повинен мати доступ до створення, редагування й видалення лише власних оголошень. Такий підхід відповідає принципу контролю доступу, згідно з яким користувачу слід надавати тільки ті дозволи, які потрібні для виконання його дій. OWASP описує принцип найменших привілеїв як підхід, за якого після автентифікації та авторизації доступ користувача обмежується мінімально необхідними правами.[25]

№	Нефункціональна вимога	Опис
НФВ-1	Адаптивність	Інтерфейс повинен коректно відображатися на смартфонах, планшетах і комп'ютерах.
НФВ-2	Зручність використання	Основні дії користувача мають бути зрозумілими без додаткового навчання.
НФВ-3	Безпека	Паролі не повинні зберігатися у відкритому вигляді; дані користувачів мають бути захищені від несанкціонованих дій.
НФВ-4	Надійність	Система повинна коректно обробляти помилки, порожні запити, неправильні дані та недоступність окремих ресурсів.
НФВ-5	Продуктивність	Каталог оголошень і фільтрація повинні працювати без помітних затримок для користувача.
НФВ-6	Підтримуваність	Код застосунку має бути структурований так, щоб його можна було розширювати та змінювати.
НФВ-7	Сумісність	Застосунок повинен працювати у сучасних браузерях без необхідності встановлення додаткового програмного забезпечення.
НФВ-8	Доступність	Інтерфейс має бути зрозумілим, читабельним і придатним для користування на різних пристроях.

Таблиця 2.2 — Нefункціональні вимоги до застосунку

Адаптивність є однією з ключових нефункціональних вимог, оскільки застосунок розробляється саме у форматі мобільного вебзастосунку. MDN

Web Docs зазначає, що адаптивний дизайн може використовувати гнучкі сітки, відносні одиниці вимірювання, мінімальні та максимальні значення розмірів, а також медіазапити для коректного відображення сторінок на різних екранах.[35] Bootstrap 5 також підтримує адаптивну сітку з контрольними точками для різних ширин екрана, зокрема sm, md, lg, xl і xxl.[9]

Зручність використання застосунку має спиратися на загальні принципи UX/UI. Nielsen Norman Group виділяє десять евристик зручності інтерфейсу, серед яких видимість стану системи, відповідність між системою і реальним світом, контроль користувача, послідовність, попередження помилок і мінімалістичний дизайн.[31] Для «BookShelf» це означає, що кнопки створення оголошення, пошуку, фільтрації, редагування і видалення мають бути помітними, а повідомлення про помилки — зрозумілими для користувача.

Окремо потрібно враховувати доступність вебінтерфейсу. У рекомендаціях WCAG 2.2 зазначено, що вебконтент має будуватися на чотирьох принципах: сприйнятність, керованість, зрозумілість і надійність.[40] Для проєкту це означає, що текст має бути достатньо читабельним, елементи керування — помітними, а форми — зрозумілими та логічно підписаними.

Безпека системи є важливою через наявність реєстрації, входу та збереження користувацьких даних. OWASP рекомендує не зберігати паролі у відкритому вигляді, а використовувати безпечні методи зберігання паролів, щоб навіть у разі компрометації бази даних зменшити ризик отримання реальних паролів користувачів.[34] Також OWASP рекомендує перевіряти вхідні дані, оскільки валідація введення є базовим механізмом захисту застосунку від некоректних або небезпечних даних.[22]

2.2. Сценарії варіантів використання застосунку

Для опису взаємодії користувачів із системою застосовуються сценарії варіантів використання. UML є стандартною мовою моделювання, призначеною для візуалізації, специфікації, конструювання та

документування артефактів програмних систем.[7] У межах цієї роботи сценарії використання застосовуються для опису основних дій користувачів у мобільному вебзастосунку «BookShelf».

У системі можна виділити два основні типи акторів:

1. Гість — користувач, який ще не виконав вхід у систему.
2. Авторизований користувач — користувач, який зареєструвався та увійшов у систему.

Гість може переглядати каталог, шукати книги, використовувати фільтри, відкривати детальну інформацію про оголошення, а також перейти до реєстрації або входу. Авторизований користувач має ширші можливості: створення оголошень, редагування й видалення власних оголошень, перегляд профілю та зв'язок із продавцем. Основні варіанти використання можна згрупувати у таблицю, приклад наведений у Таблиці 2.3

№	Варіант використання	Актор	Короткий опис
UC-1	Реєстрація	Гість	Користувач створює обліковий запис у системі.
UC-2	Вхід у систему	Гість	Користувач вводить email і пароль для авторизації.
UC-3	Перегляд каталогу	Гість / авторизований користувач	Користувач переглядає список оголошень про книги.
UC-4	Пошук книги	Гість / авторизований користувач	Користувач шукає книгу за назвою.
UC-5	Фільтрація оголошень	Гість / авторизований користувач	Користувач фільтрує оголошення за містом або типом.
UC-6	Перегляд деталей книги	Гість / авторизований користувач	Користувач відкриває докладну інформацію про оголошення.
UC-7	Створення оголошення	Авторизований користувач	Користувач додає нову книгу для продажу або обміну.
UC-8	Редагування оголошення	Авторизований користувач	Користувач змінює дані власного оголошення.

UC-9	Видалення оголошення	Авторизований користувач	Користувач видаляє власне оголошення.
UC-10	Перегляд профілю	Авторизований користувач	Користувач переглядає власну інформацію та свої оголошення.
UC-11	Зв'язок із продавцем	Гість / авторизований користувач	Користувач переходить до написання листа продавцю через email.
UC-12	Вихід із системи	Авторизований користувач	Користувач завершує роботу зі своїм обліковим записом.

Таблиця 2.3 — Основні варіанти використання

2.3. Проектування архітектури застосунку

Архітектура мобільного вебзастосунку «BookShelf» проектується за клієнт-серверним принципом. У такій моделі клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем, серверна частина — за обробку запитів, бізнес-логіку та роботу з базою даних. MDN Web Docs описує HTTP як протокол, у якому клієнт і сервер обмінюються окремими повідомленнями: клієнт надсилає запити, а сервер повертає відповіді.[33]

Загальна архітектура системи складається з таких частин:

1. Клієнтська частина — сторінки мобільного вебзастосунку, форми, каталог книг, профіль користувача, елементи пошуку й фільтрації.
2. Серверна частина — API, яке приймає HTTP-запити, обробляє дані, перевіряє права користувача та взаємодіє з базою даних.
3. База даних — зберігає інформацію про користувачів і книжкові оголошення.
4. Механізм авторизації — забезпечує доступ до функцій, які доступні лише зареєстрованим користувачам.

Загальну архітектуру інформаційної системи BookShelf зображено на рисунку 2.1.

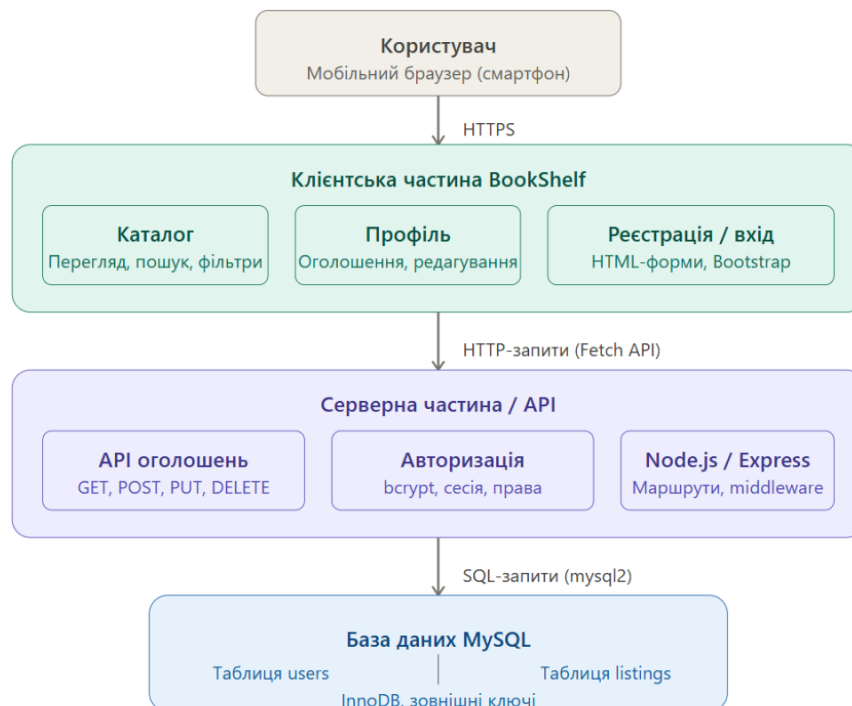


Рисунок 2.1 — Загальна архітектура мобільного вебзастосунку «BookShelf»

Клієнтська частина надсилає запити до серверної частини за допомогою HTTP. MDN зазначає, що HTTP працює за класичною клієнт-серверною моделлю: клієнт відкриває з'єднання, надсилає запит і очікує відповідь від сервера.[18] Для взаємодії JavaScript-коду з сервером може використовуватися Fetch API, який надає інтерфейс для отримання ресурсів і виконання мережових запитів.[16]

Серверна частина може бути реалізована на Node.js з використанням Express.js. Node.js є асинхронним подієво-орієнтованим середовищем виконання JavaScript, призначеним для створення масштабованих мережових застосунків.[32] Express.js є легким і гнучким фреймворком для маршрутизації та роботи з middleware у Node.js-застосунках.[12]

Логіку системи доцільно розділити на такі рівні:

- рівень інтерфейсу — HTML-сторінки, стилі, JavaScript-обробники;
- рівень API — маршрути для реєстрації, входу, отримання, створення, редагування та видалення оголошень;

- рівень бізнес-логіки — перевірка даних, перевірка прав користувача, підготовка відповідей;
- рівень даних — SQL-запити до бази даних.

Такий поділ спрощує підтримку застосунку. Наприклад, зміни в інтерфейсі не повинні вимагати повної зміни структури бази даних, а зміни в API мають бути контрольованими й не порушувати основні сценарії використання.

Безпека архітектури має включати:

- хешування паролів;
- перевірку вхідних даних;
- перевірку прав доступу до оголошень;
- використання параметризованих SQL-запитів;
- обмеження повернення зайвих персональних даних через API.

OWASP рекомендує використовувати prepared statements із параметризованими запитам для запобігання SQL-ін'єкціям, оскільки такий підхід відокремлює SQL-код від даних користувача. Це особливо важливо для системи, де користувач вводить email, пароль, назву книги, опис і примітки до оголошення.[36]

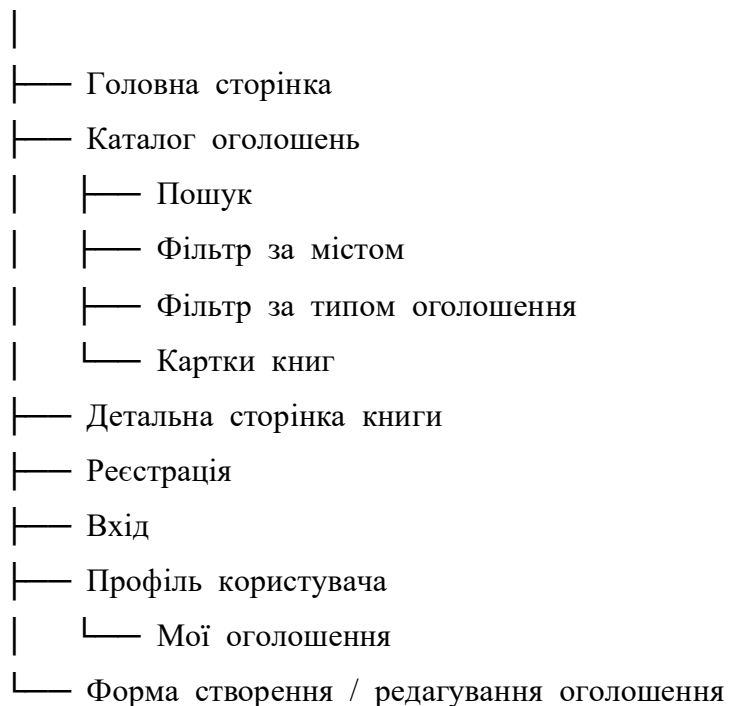
Проектування архітектурного каркасу мобільного вебзастосунку

Архітектурний каркас мобільного вебзастосунку «BookShelf» визначає основні сторінки, модулі та зв'язки між ними. Оскільки застосунок орієнтований на мобільне використання, структура сторінок повинна бути простою, а основні дії мають виконуватися за мінімальну кількість кроків.

Під час проектування інтерфейсу використовується підхід адаптивної верстки. MDN зазначає, що медіазапити дозволяють застосовувати CSS-правила залежно від характеристик пристрою, зокрема ширини viewport, і є важливою частиною responsive web design.[26] Bootstrap 5 підтримує адаптивну grid-систему, у якій контрольні точки базуються на min-width media queries, що дозволяє керувати поведінкою контейнерів і колонок на різних розмірах екрана.[17]

Структура клієнтської частини може бути подана так:

Клієнтська частина



Основним елементом інтерфейсу є картка книги. Вона повинна містити стислу інформацію, достатню для первинного вибору: фото, назву, стан, тип оголошення, місто та ціну або інформацію про обмін.

Навігація в мобільному вебзастосунку повинна бути простою. Основні розділи — каталог, створення оголошення, профіль і вхід/вихід — мають бути доступні з верхнього меню або компактної мобільної навігації. Для мобільного формату важливо уникати складних багаторівневих меню, оскільки на малих екранах вони ускладнюють взаємодію.

Каркас застосунку повинен також враховувати стани інтерфейсу:

- користувач не авторизований;
- користувач авторизований;
- каталог завантажується;
- оголошень не знайдено;
- форма заповнена з помилками;
- операція виконана успішно;
- сервер повернув помилку.

Відображення стану системи є однією з евристик Nielsen Norman Group:

користувач має розуміти, що відбувається після натискання кнопки, надсилання форми або виникнення помилки.[31]

2.4. Проєктування бази даних та API для взаємодії клієнтської і серверної частин

Для зберігання даних у системі «BookShelf» доцільно використати реляційну базу даних. Основними сутностями системи є користувач і оголошення. Користувач може створювати багато оголошень, але кожне оголошення належить одному користувачу. Такий зв'язок відповідає моделі «один-до-багатьох».

MySQL підтримує первинні ключі, які використовуються для унікальної ідентифікації записів у таблиці.

Для зв'язку таблиць доцільно використовувати зовнішні ключі. MySQL визначає зовнішній ключ як зв'язок між батьківською таблицею, що містить початкові значення, і дочірньою таблицею, яка посилається на ці значення; такі обмеження допомагають підтримувати узгодженість пов'язаних даних.[30]

Проєктування таблиці користувачів

Таблиця users призначена для зберігання облікових записів користувачів наведено в Таблиці 2.4

№	Назва поля	Тип даних	Обмеження	Опис
1	id	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор користувача
2	name	VARCHAR(100)	NOT NULL	Ім'я користувача (нікнейм)
3	email	VARCHAR(150)	UNIQUE, NOT NULL	Електронна пошта для входу та зв'язку
4	password	VARCHAR(255)	NOT NULL	Хешований пароль (bcrypt)
5	city	VARCHAR(100)	NULL	Місто проживання користувача
6	created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата та час реєстрації

Таблиця 2.4 — Структура таблиці users

Зберігання пароля у вигляді хешованого значення в полі password, а не відкритого тексту, є обов'язковою вимогою безпеки. OWASP підкреслює, що паролі мають бути захищені навіть у разі компрометації бази даних, а сучасні мови й фреймворки надають засоби для безпечного зберігання паролів.[34]

Проектування таблиці оголошень

Таблиця listings призначена для зберігання оголошень про продаж або обмін книг наведено в Таблиці 2.5

№	Назва поля	Тип даних	Обмеження	Опис
1	id	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор оголошення
2	user_id	INT	FOREIGN KEY → users(id), NOT NULL	Посилання на користувача-власника
3	title	VARCHAR(200)	NOT NULL	Назва книги
4	author	VARCHAR(150)	NULL	Автор книги
5	description	TEXT	NULL	Опис книги та стан примірника
6	type	ENUM	('sale','exchange'), NOT NULL	Тип оголошення: продаж або обмін
7	condition_book	VARCHAR(50)	NULL	Стан книги (нова, вживана тощо)
8	price	DECIMAL(10,2)	NULL	Ціна (NULL для оголошень обміну)
9	city	VARCHAR(100)	NOT NULL	Місто, де знаходиться книга
10	image_url	VARCHAR(500)	NULL	Шлях до зображення книги
11	seller_note	TEXT	NULL	Додаткова нотатка продавця
12	created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата та час створення оголошення

Таблиця 2.5 — Структура таблиці listings

Зв'язок між таблицями users і listings реалізується через поле user_id. Це дозволяє визначити власника кожного оголошення і перевіряти права доступу під час редагування або видалення. Для підтримки цілісності даних доцільно

використовувати обмеження зовнішнього ключа, оскільки MySQL підтримує foreign key constraints для узгодженості пов'язаних таблиць.[30]

Проектування API

API забезпечує взаємодію між клієнтською та серверною частинами. Клієнтська частина надсилає HTTP-запити, сервер обробляє їх, звертається до бази даних і повертає відповідь у зручному форматі. MDN зазначає, що HTTP-статуси відповіді показують, чи був конкретний HTTP-запит успішно завершений, і поділяються на групи: інформаційні, успішні, перенаправлення, помилки клієнта та помилки сервера.[20]

Для «BookShelf» доцільно використати REST-підхід до API. Microsoft у рекомендаціях щодо RESTful Web API зазначає, що найпоширенішими HTTP-методами є GET, POST, PUT, PATCH і DELETE, а кожен метод має відповідати певній операції над ресурсом.[39] MDN також пояснює, що HTTP-методи визначають призначення запиту та очікувану дію у разі успішного виконання.[19]

Основні API-маршрути застосунку наведено в таблиці 2.6.

Метод	Маршрут	Призначення
POST	/api/register	Реєстрація нового користувача
POST	/api/login	Авторизація користувача
GET	/api/listings	Отримання списку всіх оголошень
GET	/api/listings/:id	Отримання одного оголошення за ідентифікатором
POST	/api/listings	Створення нового оголошення
PUT	/api/listings/:id	Оновлення власного оголошення
DELETE	/api/listings/:id	Видалення власного оголошення
GET	/api/my-listings/:userId	Отримання оголошень конкретного користувача
PUT	/api/users/:id	Оновлення даних профілю користувача

Таблиця 2.6 — Перелік маршрутів API застосунку BookShelf

Перевірка даних і безпека API

Оскільки користувач вводить дані через форми, сервер повинен перевіряти всі отримані значення. OWASP визначає валідацію введення як набір технік, які гарантують, що до програмного компонента потрапляють тільки коректно сформовані дані.[38] Для «BookShelf» це означає перевірку email, обов'язкових полів, типу оголошення, числових значень ціни та довжини текстових полів.

Для SQL-запитів потрібно використовувати параметризовані запити. OWASP зазначає, що prepared statements із parameterized queries є основним способом запобігання SQL-ін'єкціям, оскільки SQL-код задається окремо від параметрів користувача.[36]

Для маршрутів редагування й видалення оголошень необхідно виконувати перевірку власника ресурсу. OWASP рекомендує застосовувати підхід deny-by-default, тобто за замовчуванням відмовляти в доступі, якщо немає явного дозволу на виконання дії.[8] У контексті «BookShelf» це означає, що користувач не може змінити або видалити оголошення, якщо його user_id не збігається з ідентифікатором власника оголошення.

Висновки до розділу 2

У другому розділі було виконано проєктування мобільного вебзастосунку «BookShelf». Спочатку визначено функціональні вимоги до системи, серед яких реєстрація, вхід, перегляд каталогу, пошук, фільтрація, створення, редагування та видалення оголошень, перегляд профілю і зв'язок із продавцем через email.

Також було сформульовано нефункціональні вимоги, зокрема адаптивність, зручність використання, безпеку, надійність, продуктивність, підтримуваність, сумісність і доступність. Ці вимоги базуються на загальних підходах до якості програмного забезпечення, UX/UI-дизайну, адаптивної верстки та безпеки вебзастосунків.

У розділі описано основні сценарії використання системи для гостя та

авторизованого користувача. Було визначено, що гість може переглядати каталог, шукати й фільтрувати книги, а авторизований користувач додатково може створювати, редагувати та видаляти власні оголошення.

Архітектуру застосунку спроектовано за клієнт-серверним принципом. Клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, серверна частина — за обробку запитів і бізнес-логіку, а база даних — за зберігання інформації про користувачів та оголошення.

Також було спроектовано базу даних із двома основними сутностями: користувачі та оголошення. Для взаємодії клієнтської і серверної частин визначено REST API-маршрути, які забезпечують реєстрацію, авторизацію, отримання, створення, оновлення та видалення оголошень. Особливу увагу приділено перевірці даних, захисту паролів, параметризованим SQL-запитам і контролю доступу до власних оголошень.

Результати цього розділу створюють основу для подальшої реалізації мобільного вебзастосунку «BookShelf» у наступному розділі кваліфікаційної роботи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОБІЛЬНОГО ВЕБЗАСТОСУНКУ «BookShelf»

3.1. Вибір технологій для розробки застосунку

Для реалізації BookShelf було використано стек, який відповідає архітектурі невеликого клієнт-серверного вебзастосунку: HTML, CSS, Bootstrap і JavaScript для клієнтської частини, Node.js та Express.js для серверної логіки, MySQL для зберігання даних. Такий вибір дозволив реалізувати каталог оголошень, авторизацію користувачів, профіль, CRUD-операції з оголошеннями та взаємодію між фронтом і API.

HTML використано для побудови сторінок index.html, catalog.html і profile.html. Ці сторінки відповідають основним сценаріям роботи: перегляду головної сторінки, пошуку в каталозі та керуванню власним профілем.

CSS і Bootstrap 5 застосовано для оформлення інтерфейсу, побудови сітки сторінок, стилізації карток книг, форм, кнопок, модальних вікон і адаптації макета до різних розмірів екрана. Документація Bootstrap описує grid-систему як набір контейнерів, рядків і колонок для створення гнучких макетів.[17]

JavaScript використано не лише для окремих інтерактивних елементів, а як основний шар клієнтської логіки. У файлі assets/js/app.js реалізовано завантаження оголошень із сервера, формування HTML-карток, роботу фільтрів, відкриття модальних вікон, створення й редагування оголошень, обробку авторизації та виведення повідомлень користувачу.

Node.js обрано як середовище виконання JavaScript на сервері. Це дало змогу використовувати одну мову програмування на клієнтській і серверній частинах, що спрощує розробку та підтримку навчального проєкту. У документації Node.js зазначено, що середовище орієнтоване на створення масштабованих мережевих застосунків.[32]

Express.js використано для створення HTTP-маршрутів і підключення middleware. У проєкті серверний файл server.js не містить усю бізнес-логіку, а виконує роль точки входу: підключає CORS, JSON-обробник, з'єднання з

базою даних і окремі роутери `authRoutes` та `listingRoutes`.

MySQL використано для зберігання структурованих даних. У файлі `database.sql` створюється база `book_app` і дві основні таблиці: `users` для облікових записів та `listings` для книжкових оголошень. Зв'язок між ними реалізовано через поле `user_id` з обмеженням зовнішнього ключа.

Бібліотека `mysql2` забезпечує підключення Node.js-застосунку до MySQL. Параметри підключення винесено у змінні середовища `DB_HOST`, `DB_USER`, `DB_PASSWORD` і `DB_NAME`, які зчитуються через `dotenv`.

Для захисту паролів використано `bcrypt`. Під час реєстрації пароль хешується перед збереженням у базі даних, а під час входу введений пароль порівнюється з хешованим значенням. Такий підхід відповідає рекомендаціям OWASP щодо безпечного зберігання паролів.[34]

CORS використано для коректної взаємодії клієнтської частини з сервером на `localhost:3000`. MDN визначає Cross-Origin Resource Sharing як механізм на основі HTTP-заголовків, який дозволяє серверу визначати допустимі джерела запитів.[10]

Таким чином, обрані технології не просто перелічені у роботі, а пов'язані з конкретними частинами реалізації BookShelf: сторінками інтерфейсу, файлом клієнтської логіки, серверним API, підключенням до бази даних і механізмами авторизації.

3.2. Реалізація архітектурного каркасу застосунку

Архітектурний каркас BookShelf реалізовано як набір взаємопов'язаних файлів клієнтської та серверної частин. На клієнті розміщено HTML-сторінки, стилі та файл JavaScript-логіки, а на сервері — точку входу `server.js`, конфігурацію підключення до бази даних і окремі маршрути для авторизації та оголошень.

У таблиці 3.1 наведено основні файли проєкту та їхнє призначення.

Компонент	Призначення
index.html, catalog.html, profile.html	Основні сторінки інтерфейсу: головна сторінка, каталог із фільтрами та профіль користувача.
assets/css/styles.css, Bootstrap	Стилізація, адаптивна сітка, оформлення карток, кнопок, форм і модальних вікон.
assets/js/app.js	Клієнтська логіка: стан користувача, запити до API, фільтрація, рендеринг карток, створення, редагування та видалення оголошень.
server.js	Точка входу серверної частини: запуск Express, CORS, JSON-обробник, підключення бази даних і роутерів API.
config/db.js	Підключення до MySQL через mysql2 з використанням змінних середовища.
routes/authRoutes.js	Маршрути реєстрації, входу, оновлення профілю та зміни пароля.
routes/listingRoutes.js	Маршрути отримання, пошуку, створення, редагування та видалення книжкових оголошень.
database.sql	SQL-скрипт створення бази book_app, таблиць users і listings та індексів для фільтрації.

Таблиця 3.1 — Архітектурний каркас і структура файлів проєкту

Файл `server.js` виконує роль центральної точки запуску: створює Express-застосунок, підключає `middleware cors()` та `express.json()`, перевіряє з'єднання з MySQL, визначає тестовий маршрут `/` і підключає роутери через `app.use("/api", authRoutes)` та `app.use("/api", listingRoutes)`.

Такий поділ робить структуру зрозумілішою: логіка реєстрації, входу й редагування профілю зосереджена в `routes/authRoutes.js`, а логіка створення, пошуку, редагування та видалення оголошень — у `routes/listingRoutes.js`. Express дає можливість описувати маршрути з параметрами, наприклад `:id`, що дозволяє працювати з конкретним ресурсом за його ідентифікатором.[13]

3.3. Реалізація клієнтської частини мобільного вебзастосунку

Клієнтська частина BookShelf реалізована як набір HTML-сторінок, що використовують спільні стилі та один основний файл логіки `assets/js/app.js`. Саме цей файл відповідає за обмін даними із сервером, оновлення інтерфейсу без ручного дублювання коду на кожній сторінці та контроль доступності окремих дій для користувача.

Сторінка `index.html` виконує роль головної сторінки із коротким описом сервісу та блоком нових оголошень. Сторінка `catalog.html` призначена для пошуку й фільтрації книг, а `profile.html` — для перегляду даних користувача, редагування профілю та керування власними оголошеннями.

У файлі `app.js` використовується об'єкт `state`, у якому зберігаються поточні дані клієнтської частини: список книг, активний користувач і дані профілю. Стан авторизації додатково зберігається в `localStorage` за ключем `bookshelf_current_user`, тому після оновлення сторінки користувач не втрачає інформацію про вхід у систему.

Для запитів до сервера реалізовано функцію `apiFetch(path, options)`. Вона додає базову адресу API `http://localhost:3000`, встановлює заголовок `Content-Type: application/json`, перетворює відповідь сервера у JSON і виводить повідомлення про помилку, якщо сервер повернув неуспішний HTTP-статус.

Завантаження каталогу виконує функція `loadBooks(filters)`. Вона формує параметри `title`, `city` і `type`, надсилає GET-запит на `/api/listings` і передає отримані дані у `normalizeBook()`. Ця функція приводить серверні поля `condition_book`, `seller_note`, `image_url`, `user_id` і `seller_email` до структури, зручної для відображення в інтерфейсі.

Кнопки редагування й видалення не виводяться для всіх оголошень однаково. Перед формуванням картки функція `isOwner(book)` порівнює `user_id` оголошення з `id` поточного користувача. Якщо ці значення збігаються, у шаблон картки додаються кнопки «Редагувати» та «Видалити»; якщо ні — користувач бачить лише перегляд оголошення та кнопку зв'язку з продавцем.

Основні функції клієнтської логіки `app.js` можна подати так:

- `loadSession()` — зчитує поточного користувача з `localStorage`;
- `renderAuthControls()` — змінює кнопки навігації залежно від стану авторизації;
- `loadBooks()` — отримує каталог оголошень із сервера з урахуванням фільтрів;
- `renderGrid()` — формує картки книг на головній сторінці та в каталозі;
- `handleListingSubmit()` — створює нове оголошення або оновлює наявне;
- `prepareEdit()` — завантажує оголошення, перевіряє власника й заповнює форму редагування;
- `deleteListing()` — надсилає запит на видалення власного оголошення;
- `buildSellerMailto()` — формує email-звернення до продавця з темою та текстом листа.

Лістинг 3.1 — Спрощений вигляд логіки запиту до API

```
async function apiFetch(url, options = {}) {
  const response = await fetch(url, options);
  const data = await response.json();

  if (!response.ok) {
    throw new Error(data.error || 'Помилка запиту');
  }
  return data;
}
```

Цей підхід дозволяє централізувати роботу із сервером і однаково обробляти успішні відповіді та помилки. Для користувача це проявляється у вигляді повідомлень про успішне виконання дії або про проблему, яку потрібно виправити.

Централізація запитів через `apiFetch` спрощує обробку помилок: одна й та сама логіка використовується для реєстрації, входу, завантаження каталогу, створення, редагування та видалення оголошень.

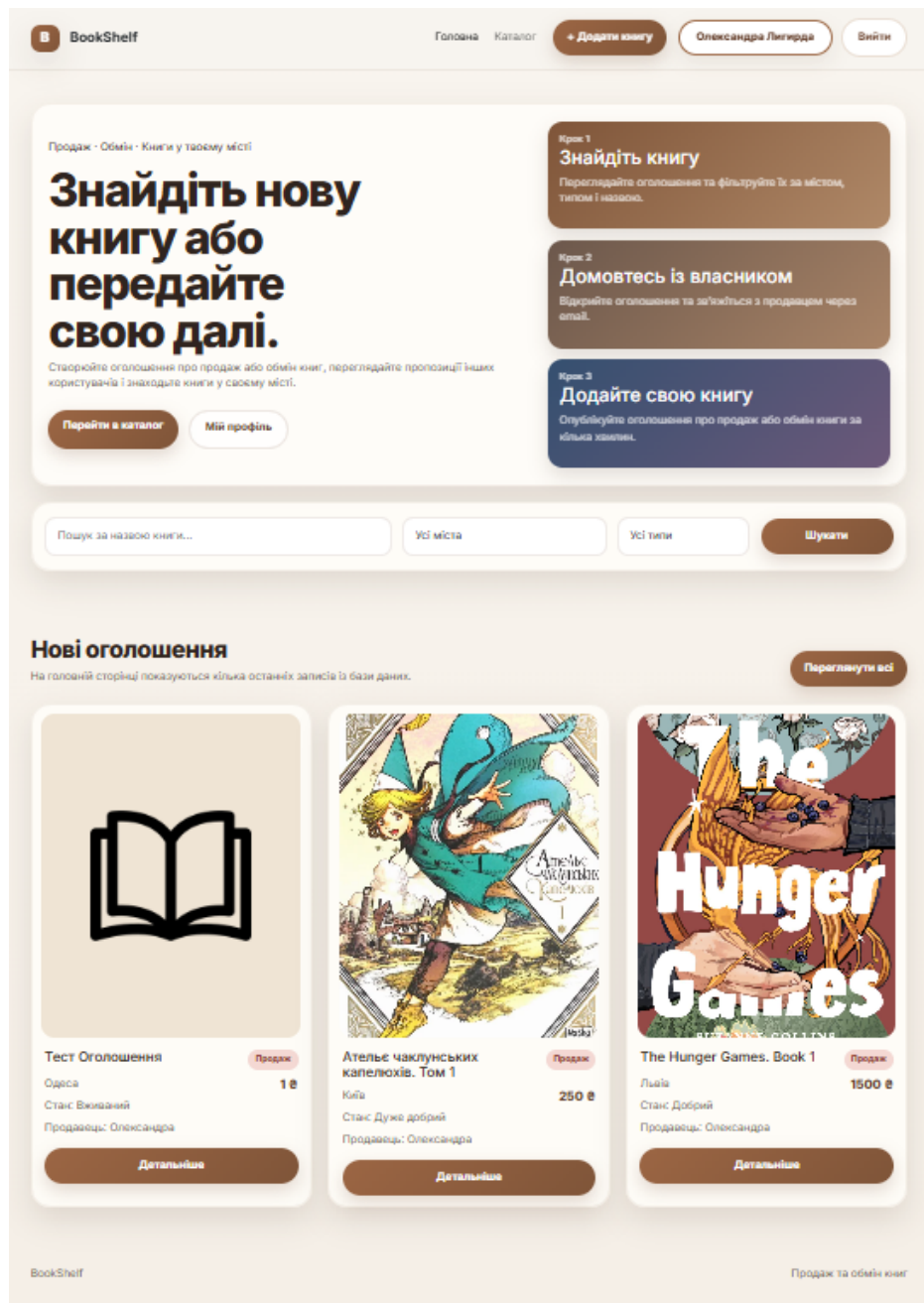


Рисунок 3.1 — Головна сторінка вебзастосунку «BookShelf»

Після входу користувача дані, отримані від `/api/login`, записуються в `state.currentUser` і `localStorage`. Після цього `renderAuthControls()` змінює навігацію: замість кнопок входу та реєстрації відображаються ім'я користувача, посилання на профіль і кнопка виходу.

Форма створення оголошення використовує ті самі поля, що й таблиця `listings` у базі даних: `title`, `description`, `condition_book`, `type`, `city`, `image_url`, `price` і `seller_note`. Якщо форма відкривається для редагування, у ній додатково

зберігається data-атрибут editId, за яким handleListingSubmit() визначає, який запит потрібно виконати: POST /api/listings або PUT /api/listings/:id.

Форми реєстрації, входу й створення оголошення реалізовані з урахуванням базової перевірки введених даних. Поля email, пароль, назва книги, місто та тип оголошення мають бути заповнені коректно. Однак клієнтська перевірка не може бути єдиним механізмом захисту, оскільки дані можуть бути надіслані напряму на сервер. OWASP наголошує, що валідація вхідних даних є важливою частиною безпеки застосунку, а отримані від користувача дані мають перевірятися перед обробкою.[38][22]

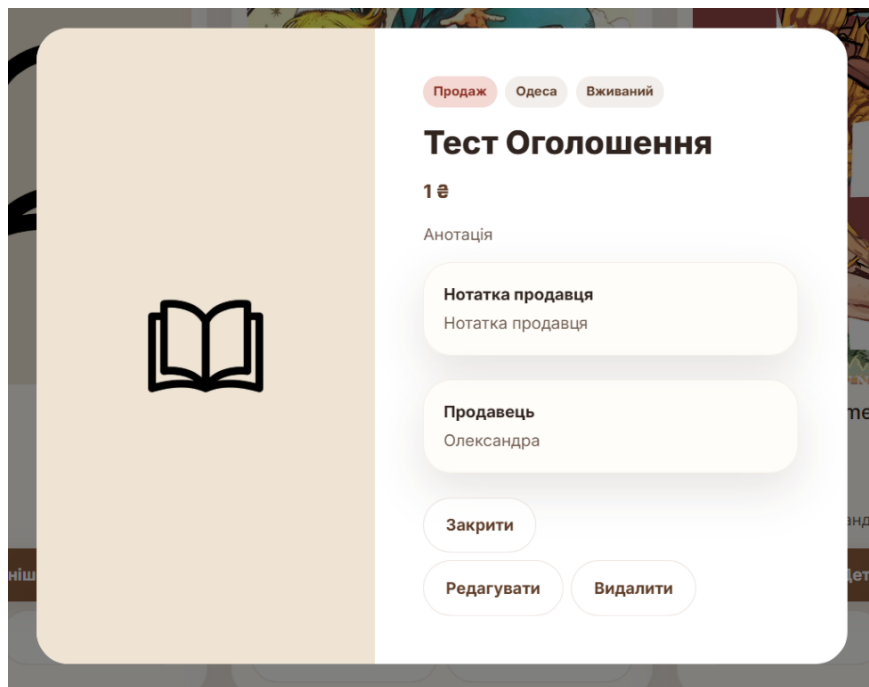
The image shows a web application interface for creating a new listing. On the left, there is a large light-brown rectangular area containing a black icon of an open book. To the right of this area is a white rounded rectangle containing the form. At the top of the form, there are three tabs: 'Продаж' (Sale) in red, 'Одеса' (Odessa) in grey, and 'Вживаний' (Active) in grey. Below the tabs is the title 'Тест Оголошення' (Test Listing) in bold. Under the title is a small icon of a person and the text '1'. Below this is the label 'Анотація' (Annotation). There are three input fields: the first is labeled 'Нотатка продавця' (Seller's note) and contains the text 'Нотатка продавця'; the second is labeled 'Продавець' (Seller) and contains the text 'Олександра'; the third is empty. At the bottom of the form are three buttons: 'Закрити' (Close), 'Редагувати' (Edit), and 'Видалити' (Delete).

Рисунок 3.2 — Форма створення нового оголошення

Таким чином, клієнтська частина не лише відображає готові сторінки, а й виконує роль посередника між користувачем і API: збирає дані з форм, перетворює їх у потрібний формат, відправляє запити на сервер і оновлює інтерфейс відповідно до результату операції.

3.4. Реалізація серверної частини, авторизації та роботи з оголошеннями

Серверна частина BookShelf реалізована на Node.js та Express.js і винесена у кілька файлів. Такий підхід є кращим за розміщення всієї логіки в

одному `server.js`, оскільки окремі групи маршрутів відповідають за різні частини системи.

Файл `server.js` створює Express-застосунок, визначає порт `PORT`, підключає `cors()`, `express.json()`, файл `config/db.js` і два роутери: `routes/authRoutes.js` та `routes/listingRoutes.js`. Після перевірки підключення до MySQL сервер запускається на `http://localhost:3000`.

Реалізація авторизації та профілю користувача

Маршрути авторизації розміщені у файлі `routes/authRoutes.js`. У ньому реалізовано `POST /api/register` для створення користувача, `POST /api/login` для входу та `PUT /api/users/:id` для оновлення профілю й зміни пароля.

Під час реєстрації сервер перевіряє наявність обов'язкових полів `name`, `email`, `password` і `city`, перевіряє мінімальну довжину пароля, шукає користувача з таким `email` і лише після цього зберігає нового користувача в таблиці `users`. Пароль перед збереженням хешується за допомогою `bcrypt`.^[34]

Лістинг 3.2 — Спрощена логіка реєстрації:

```
const passwordHash = await bcrypt.hash(password, 10);

await db.query(
  'INSERT INTO users (name, email, password, city) VALUES (?, ?, ?, ?)',
  [name, email, password, city]
);
```

У цьому фрагменті важливо, що SQL-запит виконується з параметрами, а пароль не зберігається у відкритому вигляді. Це знижує ризики, пов'язані з витоком облікових даних або некоректною обробкою введених користувачем значень.

Реалізація входу користувача

Під час входу сервер отримує `email` і `password`, знаходить користувача в таблиці `users` та порівнює введений пароль із хешованим значенням у полі `password`. У разі успіху клієнтська частина отримує об'єкт `user` з `id`, `name`, `email` і `city`.

Лістинг 3.3 — Спрощена логіка входу користувача

```
const [users] = await db.query(
  'SELECT * FROM users WHERE email = ?',
  [email]
);

const user = users[0];

const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) {
  return res.status(401).json({ error: 'Неправильний email або пароль' });
}
```

Повернення на клієнт лише потрібних полів користувача є важливим: пароль або інші службові дані не передаються у відповідь API.

Реалізація роботи з оголошеннями

Маршрути оголошень розміщені у файлі `routes/listingRoutes.js`. Вони охоплюють повний набір операцій для MVP: `POST /api/listings`, `GET /api/listings`, `GET /api/listings/:id`, `GET /api/my-listings/:userId`, `PUT /api/listings/:id` і `DELETE /api/listings/:id`.

Маршрут `GET /api/listings` підтримує параметри `title`, `city` і `type`. На сервері SQL-запит формується поступово: якщо користувач задав пошук або фільтри, до запиту додаються відповідні умови `WHERE`. Також виконується `JOIN` з таблицею `users`, щоб повернути ім'я продавця разом з оголошенням.

Під час створення оголошення `POST /api/listings` сервер перевіряє наявність `title`, `type`, `city` і `user_id`, контролює допустимі значення `type`: `sell` або `exchange`, а для оголошень продажу додатково перевіряє, що ціна більша за нуль.

Лістинг 3.4 — Структура запиту на створення оголошення:

```
{
  "title": "Назва книги",
  "description": "Опис книги",
  "condition_book": "Добрий стан",
  "type": "sell",
}
```

```

"city": "Київ",
"price": 250,
"seller_note": "Можлива зустріч у місті",
"user_id": 1
}

```

Редагування оголошення виконується через PUT /api/listings/:id. Перед оновленням сервер отримує user_id з тіла запиту, знаходить оголошення за id і порівнює user_id запису з user_id поточного користувача. Якщо значення не збігаються, сервер повертає помилку 403. Така перевірка необхідна для запобігання ситуації, коли один користувач може змінити чуже оголошення. OWASP API Security Top 10 2023 визначає Broken Object Level Authorization як одну з основних проблем API, оскільки сервер має перевіряти права доступу до кожного об'єкта, з яким працює користувач.[36]

Видалення реалізовано за тим самим принципом через DELETE /api/listings/:id. Спочатку перевіряється існування оголошення, потім власник запису, і лише після цього виконується DELETE-запит до таблиці listings. Це запобігає видаленню чужих оголошень.

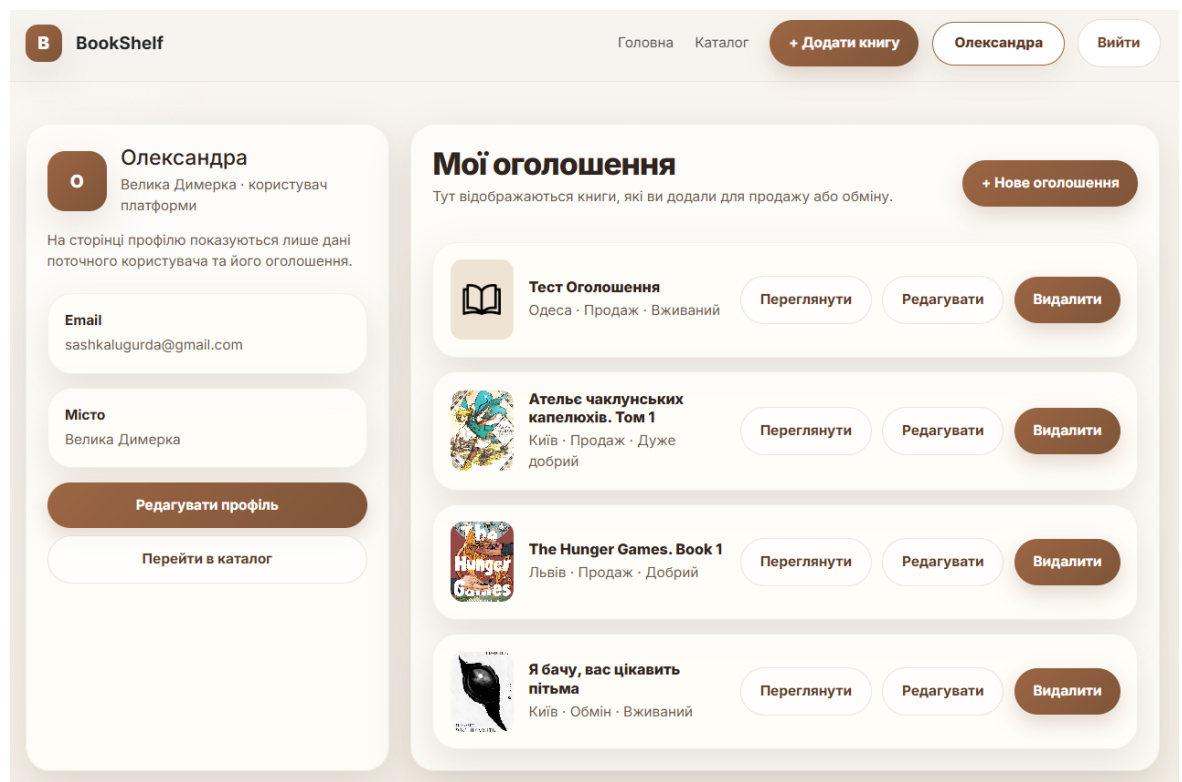


Рисунок 3.3 — Сторінка профілю користувача з власними оголошеннями

Робота з профілем і власними оголошеннями

Для сторінки профілю використовується маршрут `GET /api/my-listings/:userId`, який повертає лише оголошення конкретного користувача. Оновлення особистих даних виконується через `PUT /api/users/:id`; у цьому маршруті користувач може змінити ім'я, email, місто, а за потреби — пароль після введення поточного пароля.

Лістинг 3.5 — Приклади відповідей сервера:

```
{
  "message": "Оголошення створено успішно",
  "listingId": 1
}
{
  "message": "Немає доступу"
}
```

Сервер повертає різні HTTP-статуси залежно від ситуації: 201 для успішного створення, 400 для некоректних даних, 401 для помилки входу, 403 для забороненої дії, 404 для відсутнього користувача або оголошення та 500 для внутрішніх помилок сервера.

Забезпечення базової безпеки серверної частини

У серверній частині реалізовано кілька базових механізмів безпеки: хешування паролів, перевірку обов'язкових полів, перевірку типу оголошення, контроль ціни для продажу, параметризовані SQL-запити та перевірку власника перед редагуванням або видаленням.

Використання параметризованих SQL-запитів відповідає рекомендаціям OWASP щодо запобігання SQL-ін'єкціям, а перевірка власника оголошення зменшує ризик несанкціонованого доступу до чужих ресурсів.[36]

Отже, серверна частина BookShelf забезпечує не лише обмін даними між клієнтом і базою, а й основну бізнес-логіку застосунку: реєстрацію, вхід, оновлення профілю, пошук оголошень, створення, редагування, видалення та перевірку прав користувача.

Висновки до розділу 3

У третьому розділі описано реалізацію BookShelf як клієнт-серверного вебзастосунку. На клієнтському рівні створено сторінки `index.html`, `catalog.html` і `profile.html`, а основну логіку взаємодії з API винесено у файл `assets/js/app.js`.

У клієнтській частині реалізовано зберігання поточного користувача в `localStorage`, завантаження каталогу через `GET /api/listings`, фільтрацію за назвою, містом і типом, формування карток книг, відкриття детальної інформації та відображення кнопок редагування й видалення лише для власника оголошення.

Серверна частина організована модульно: `server.js` відповідає за запуск застосунку та підключення `middleware`, `config/db.js` — за з'єднання з MySQL, `routes/authRoutes.js` — за реєстрацію, вхід і оновлення профілю, а `routes/listingRoutes.js` — за операції з оголошеннями.

База даних `book_app` містить таблиці `users` і `listings`, пов'язані через `user_id`. У реалізації використано хешування паролів за допомогою `bcrypt`, параметризовані SQL-запити, перевірку обов'язкових полів і контроль доступу до власних оголошень.

Отримана реалізація забезпечує ключові сценарії MVP: реєстрацію, вхід, перегляд каталогу, пошук, фільтрацію, створення, редагування та видалення власних оголошень, перегляд профілю й email-зв'язок із продавцем. Це створює підстави для подальшого функціонального тестування системи.

РОЗДІЛ 4. ТЕСТУВАННЯ МОБІЛЬНОГО ВЕБЗАСТОСУНКУ

4.1. Методика і проведення тестування застосунку

Тестування є важливим етапом розробки програмного забезпечення, оскільки дозволяє перевірити відповідність реалізованого застосунку поставленим вимогам, виявити помилки та оцінити стабільність роботи основних функцій.

Для мобільного вебзастосунку «BookShelf» було проведено ручне функціональне тестування, перевірку коректності роботи форм, базову перевірку адаптивності інтерфейсу, тестування взаємодії клієнтської та серверної частин, а також перевірку обробки помилок. Основна увага приділялася тим функціям, які є ключовими для системи продажу та обміну книг: реєстрації, входу, перегляду каталогу, пошуку, фільтрації, створенню, редагуванню та видаленню оголошень.

Оскільки застосунок працює за клієнт-серверною архітектурою, тестування охоплювало не лише зовнішній вигляд сторінок, а й перевірку HTTP-запитів до серверної частини. HTTP-статуси відповіді дають змогу визначити, чи був запит успішно виконаний, чи виникла помилка клієнта або сервера. MDN поділяє HTTP-статуси на інформаційні відповіді, успішні відповіді, перенаправлення, помилки клієнта та помилки сервера.[20]

Під час тестування враховувалися такі критерії:

- коректність роботи основних функцій застосунку;
- правильність відображення сторінок на екрані комп'ютера та мобільного пристрою;
- коректна обробка введених користувачем даних;
- правильність роботи авторизації;
- можливість створення, редагування та видалення лише власних оголошень;
- робота пошуку та фільтрації;
- відображення повідомлень про успішні дії та помилки;
- стабільність взаємодії клієнтської частини з сервером і базою даних.

Для перевірки безпеки було враховано базові рекомендації OWASP. Зокрема, OWASP Web Security Testing Guide містить окремі напрями тестування автентифікації, авторизації, валідації введення та обробки помилок.[22] Для застосунку «BookShelf» це означає, що потрібно перевірити: чи не можна увійти з неправильним паролем, чи не можна створити оголошення без обов'язкових даних, чи не може користувач редагувати або видаляти чужі оголошення.

Тестування проводилося за такими напрямками:

- Функціональне тестування — перевірка основних можливостей застосунку: реєстрації, входу, перегляду каталогу, створення, редагування та видалення оголошень.
- Тестування інтерфейсу — перевірка зрозумілості розміщення елементів, кнопок, форм, карток книг і повідомлень користувачу.
- Тестування адаптивності — перевірка відображення сторінок на різних розмірах екрана. Для цього застосунок переглядався у звичайному режимі браузера та в режимі адаптивного перегляду через інструменти розробника.
- Тестування API — перевірка взаємодії клієнтської частини з сервером через маршрути API.
- Тестування обробки помилок — перевірка ситуацій, коли користувач вводить неправильні дані, залишає обов'язкові поля порожніми або намагається виконати недозволену дію.

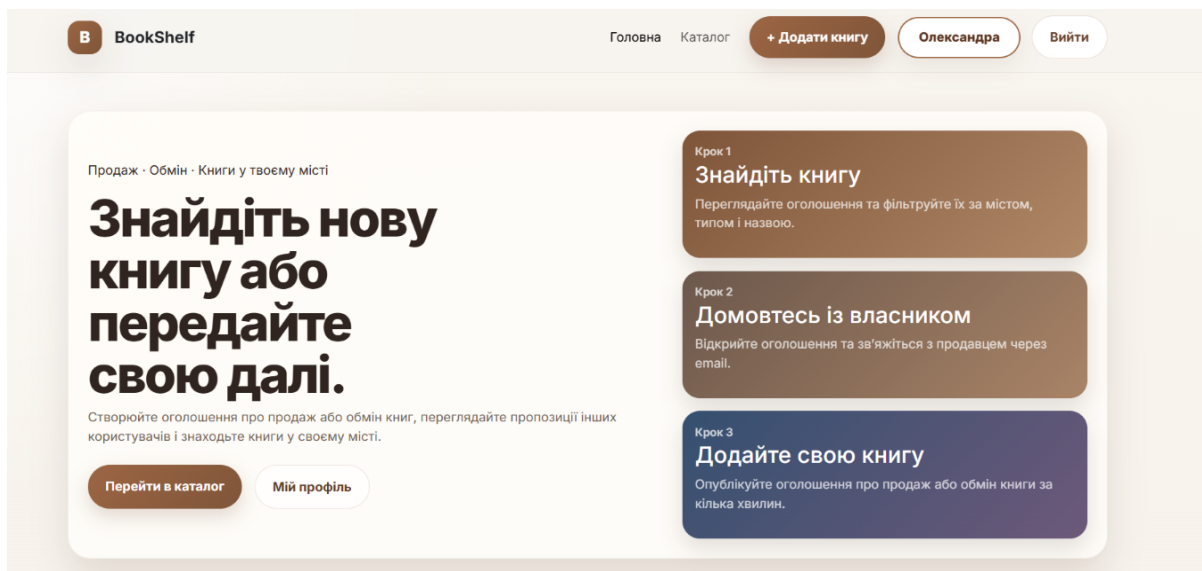


Рисунок 4.1 — Перевірка запуску мобільного вебзастосунку «BookShelf» у браузері

4.2. Результати тестування основних функціональних можливостей

Результати тестування основних функцій мобільного вебзастосунку «BookShelf» подано у таблиці 4.1. Для кожного тестового випадку визначено дію користувача, очікуваний результат, фактичний результат і статус виконання.

Під час тестування особливу увагу було приділено формам реєстрації, входу та створення оголошень. Це пов'язано з тим, що саме через форми користувач передає дані до системи. OWASP Web Security Testing Guide окремо виділяє тестування введення даних, оскільки неправильна або недостатня перевірка введення може призводити до помилок у роботі вебзастосунку.[42]

Таблиця 4.1 — Результати функціонального тестування застосунку «BookShelf»

№	Назва тесту	Дії користувача	Очікуваний результат	Фактичний результат	Статус
ТС-1	Запуск застосунку	Відкрити головну сторінку застосунку в браузері	Сторінка відкривається без помилок, відображається основний інтерфейс	Головна сторінка відкривається, елементи інтерфейсу відображаються коректно	Пройдено
ТС-2	Перехід до каталогу	Натиснути кнопку або посилання переходу до каталогу	Відкривається сторінка зі списком оголошень	Каталог завантажується, картки книг відображаються	Пройдено
ТС-3	Реєстрація нового користувача	Заповнити форму реєстрації коректними даними	Користувач успішно створюється в системі	Система створює нового користувача та показує повідомлення про успішну дію	Пройдено
ТС-4	Реєстрація з порожніми полями	Надіслати форму реєстрації без обов'язкових даних	Система не створює користувача та показує повідомлення про помилку	Система повідомляє про некоректне заповнення форми	Пройдено
ТС-5	Вхід із правильними даними	Ввести email і пароль зареєстрованого користувача	Користувач входить у систему, відкривається доступ до профілю та створення оголошень	Авторизація виконується успішно	Пройдено
ТС-6	Вхід із неправильним паролем	Ввести правильний email і неправильний пароль	Система не дозволяє увійти та показує повідомлення про помилку	Вхід не виконується, користувач отримує повідомлення про неправильні дані	Пройдено
ТС-7	Перегляд каталогу оголошень	Відкрити сторінку каталогу	Користувач бачить список доступних оголошень	Оголошення відображаються у вигляді карток книг	Пройдено
ТС-8	Пошук книги за назвою	Ввести назву або частину назви книги в поле пошуку	У каталозі залишаються оголошення, які відповідають пошуковому запиту	Пошук звужує список оголошень	Пройдено
ТС-9	Фільтрація за містом	Обрати місто у фільтрі	Відображаються оголошення з вибраного міста	Система показує відповідні оголошення	Пройдено

Продовження таблиці 4.1

№	Назва тесту	Дії користувача	Очікуваний результат	Фактичний результат	Статус
ТС-10	Фільтрація за типом оголошення	Обрати тип “продаж” або “обмін”	Відображаються оголошення відповідного типу	Фільтр працює коректно	Пройдено
ТС-11	Створення нового оголошення	Авторизований користувач заповнює форму створення оголошення	Нове оголошення додається до бази даних і з'являється в каталозі	Оголошення створюється та відображається в каталозі	Пройдено
ТС-12	Створення оголошення без обов'язкових даних	Залишити обов'язкові поля порожніми та надіслати форму	Система не створює оголошення і показує помилку	Оголошення не створюється, відображається повідомлення про помилку	Пройдено
ТС-13	Перегляд власних оголошень у профілі	Увійти в систему та перейти до профілю	Користувач бачить список власних оголошень	У профілі відображаються оголошення поточного користувача	Пройдено
ТС-14	Редагування власного оголошення	Натиснути кнопку редагування, змінити дані та зберегти	Дані оголошення оновлюються	Оголошення оновлюється після збереження змін	Пройдено
ТС-15	Видалення власного оголошення	Натиснути кнопку видалення і підтвердити дію	Оголошення видалюється з каталогу та профілю	Оголошення видалюється після підтвердження	Пройдено
ТС-16	Спроба редагування чужого оголошення	Авторизований користувач намагається змінити оголошення іншого користувача	Система забороняє дію	Редагування чужого оголошення не дозволяється	Пройдено
ТС-17	Спроба видалення чужого оголошення	Авторизований користувач намагається видалити чуже оголошення	Система забороняє дію	Видалення чужого оголошення не дозволяється	Пройдено

Продовження таблиці 4.1

№	Назва тесту	Дії користувача	Очікуваний результат	Фактичний результат	Статус
ТС-18	Зв'язок із продавцем	Натиснути кнопку зв'язку з продавцем	Відкривається можливість написати лист на email продавця	Користувач переходить до написання повідомлення продавцю	Пройдено
ТС-19	Вихід із системи	Натиснути кнопку виходу	Користувач завершує сеанс, функції авторизованого користувача стають недоступними	Вихід виконується коректно	Пройдено
ТС-20	Адаптивність інтерфейсу	Відкрити застосунок на різних розмірах екрана	Елементи інтерфейсу не виходять за межі екрана, сторінки зручні для перегляду	Інтерфейс адаптується до ширини екрана	Пройдено

Рисунок 4.2 — Тестування реєстрації користувача

Після реєстрації було перевірено функцію входу. Користувач вводив email і пароль, після чого система перевіряла наявність такого користувача та правильність введеного пароля. У разі правильних даних користувач отримував доступ до особистого профілю, а у разі помилки — повідомлення про неправильні облікові дані.

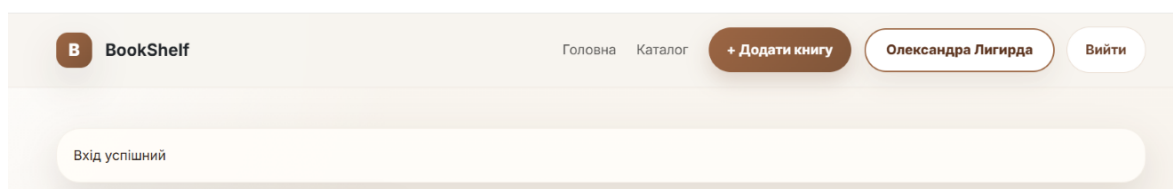


Рисунок 4.3 — Тестування входу користувача в систему

Наступним етапом було перевірено роботу каталогу оголошень. У каталозі повинні відображатися картки книг із короткою інформацією: назвою, зображенням, містом, станом, типом оголошення та коротким описом.

Також було перевірено, чи не порушується розміщення карток на різних розмірах екрана.

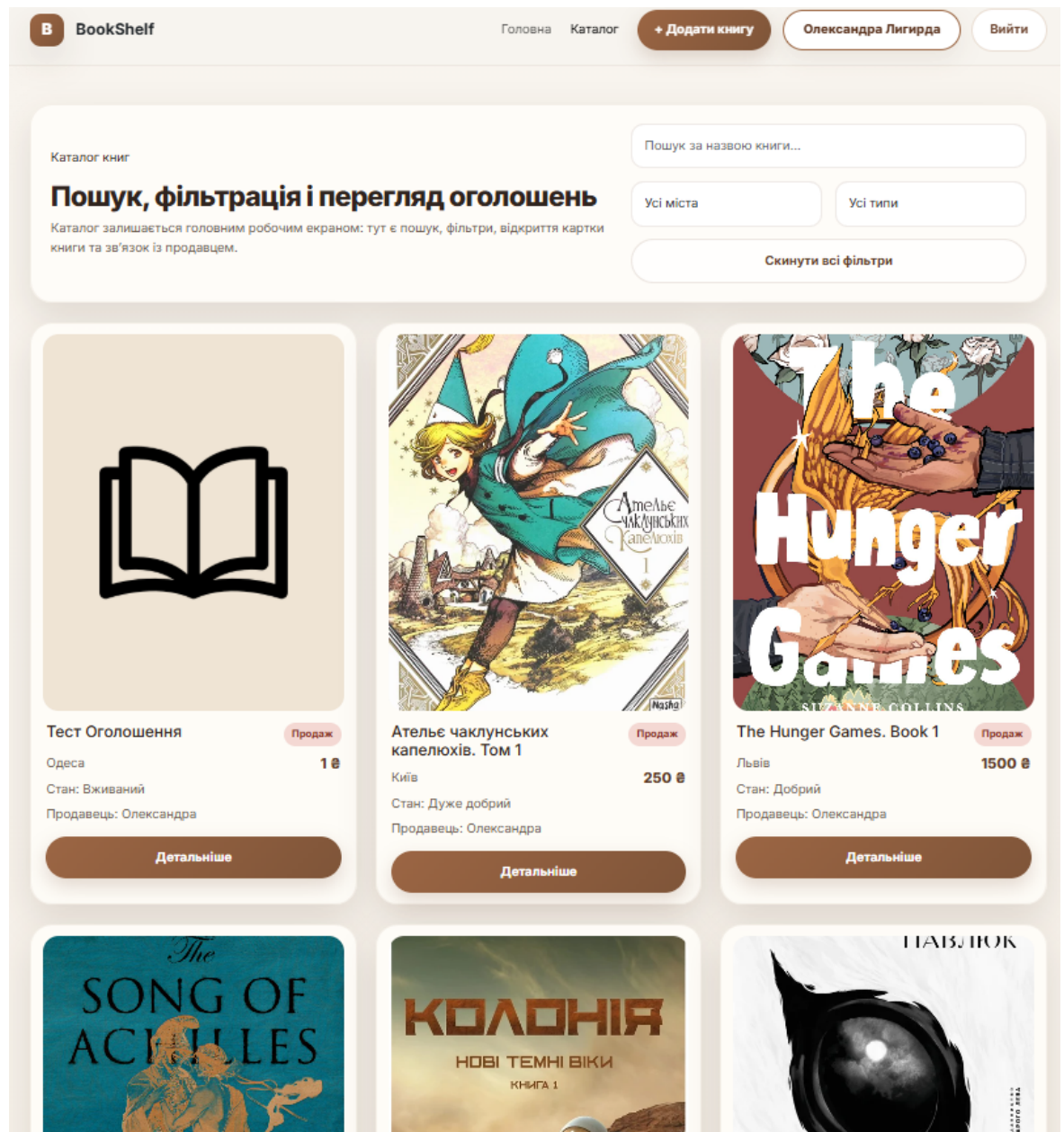


Рисунок 4.4 — Тестування відображення каталогу оголошень

Після цього було перевірено пошук і фільтрацію. Користувач вводив назву книги або її частину, а система мала залишити в каталозі лише ті оголошення, які відповідають пошуковому запиту. Також було протестовано фільтрацію за містом і типом оголошення. Така перевірка є важливою, оскільки пошук і фільтри належать до основних функцій системи та

безпосередньо впливають на зручність користування.

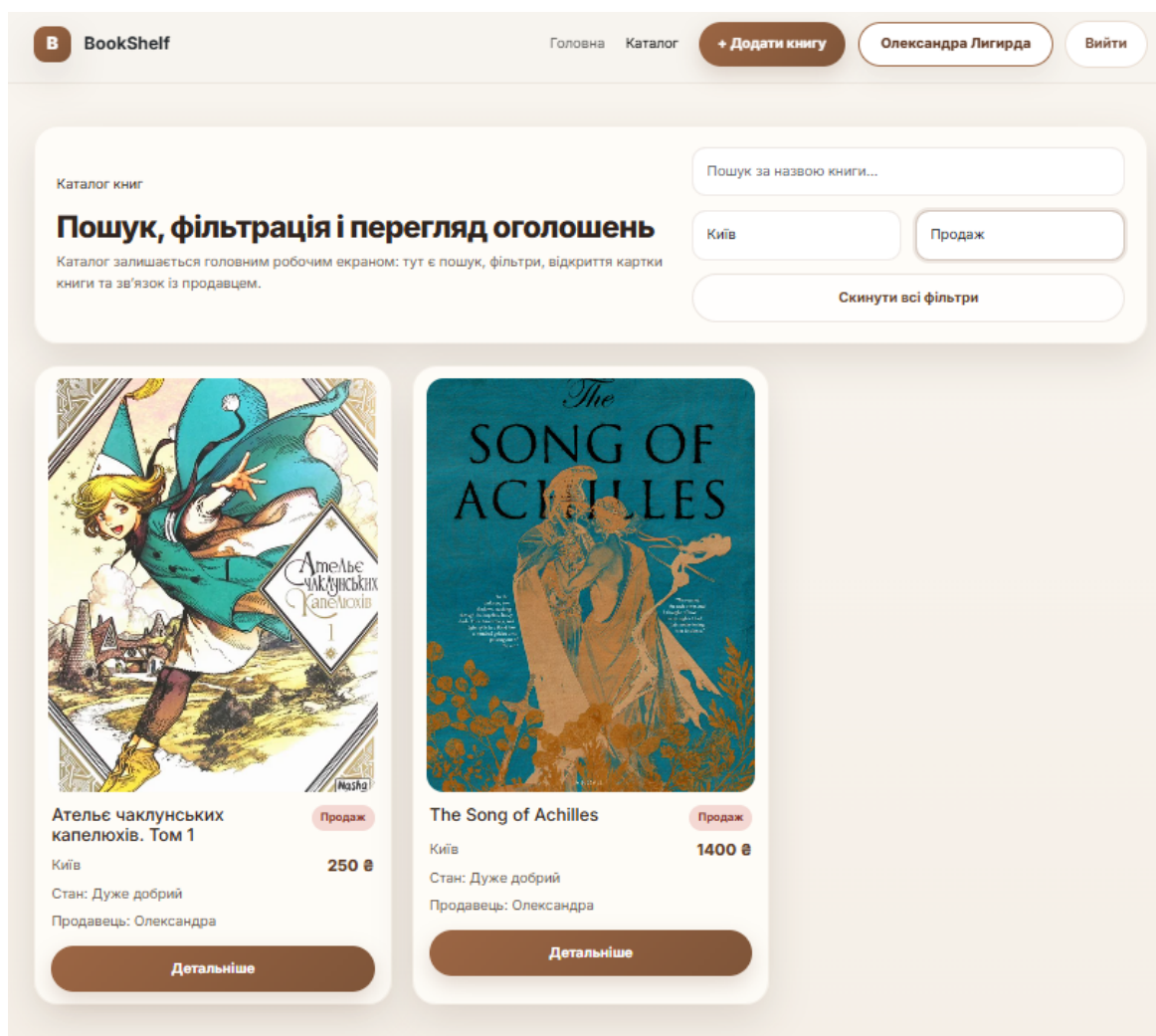


Рисунок 4.5 — Тестування пошуку та фільтрації оголошень

Окремо було перевірено створення нового оголошення. Авторизований користувач заповнював форму, вказуючи назву книги, опис, стан, тип оголошення, місто, ціну або примітку. Після надсилання форми система мала зберегти оголошення в базі даних і відобразити його в каталозі.

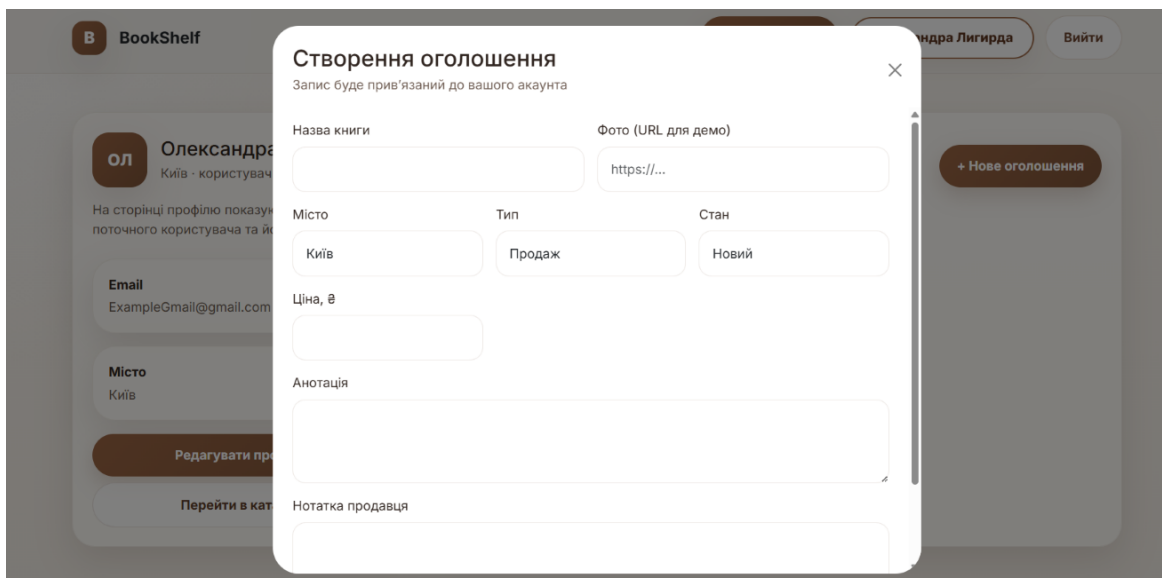


Рисунок 4.6 — Відображення створеного оголошення в системі

Також було перевірено редагування та видалення оголошень. Користувач повинен мати змогу змінювати або видаляти лише власні оголошення. Така перевірка пов'язана з авторизацією доступу до об'єктів.

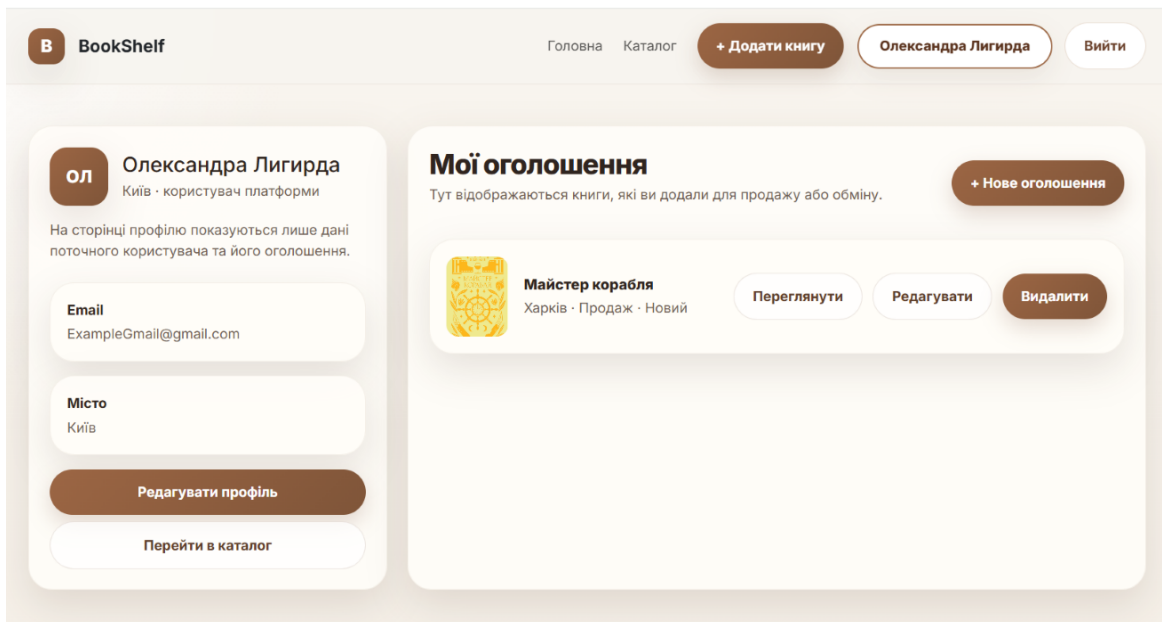


Рисунок 4.7 — Тестування редагування власного оголошення

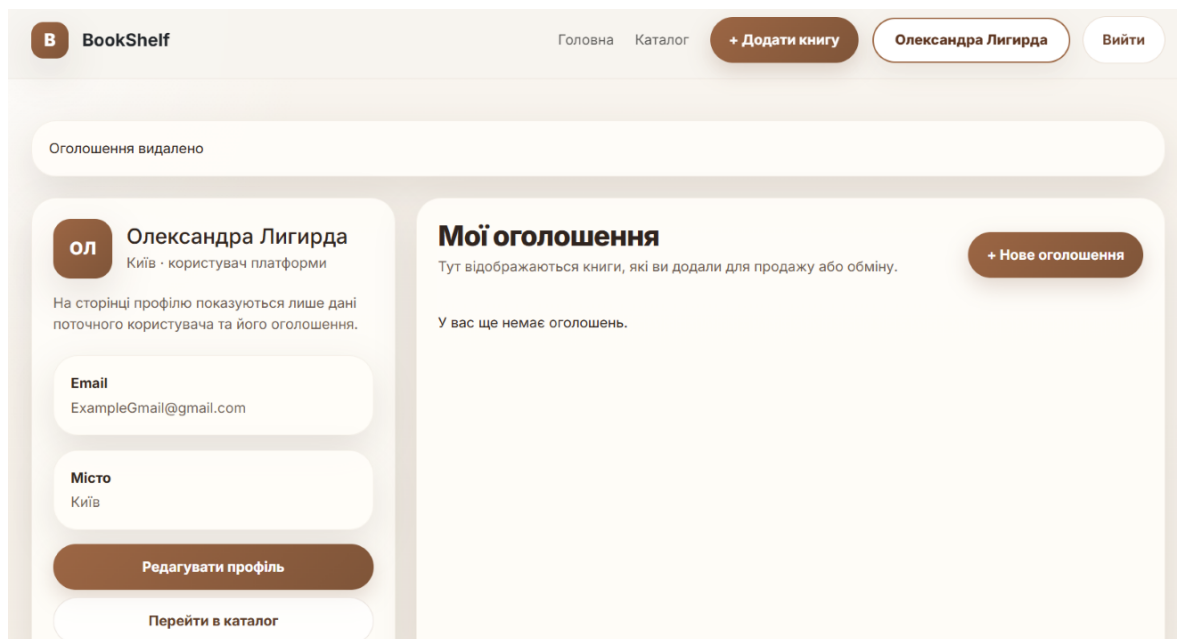


Рисунок 4.8 — Тестування видалення власного оголошення

Під час тестування обробки помилок було перевірено ситуації, коли користувач вводить неправильний пароль, залишає порожні обов'язкові поля або намагається виконати дію без авторизації. У таких випадках система повинна не виконувати дію і показувати зрозуміле повідомлення.

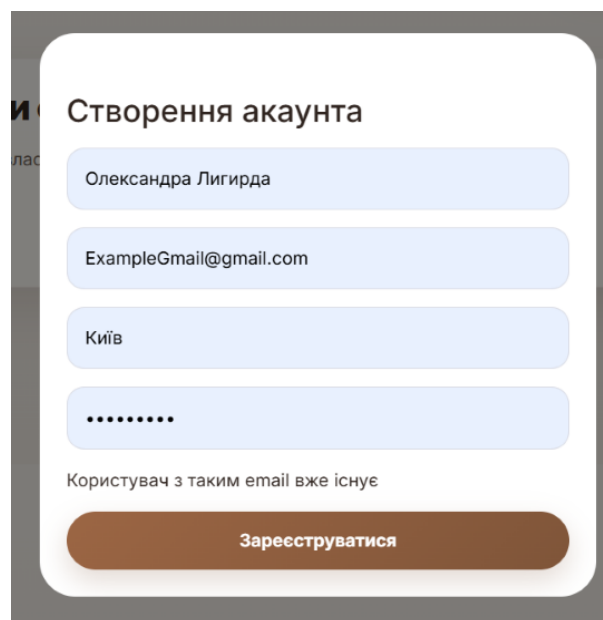


Рисунок 4.9 — Тестування обробки помилок

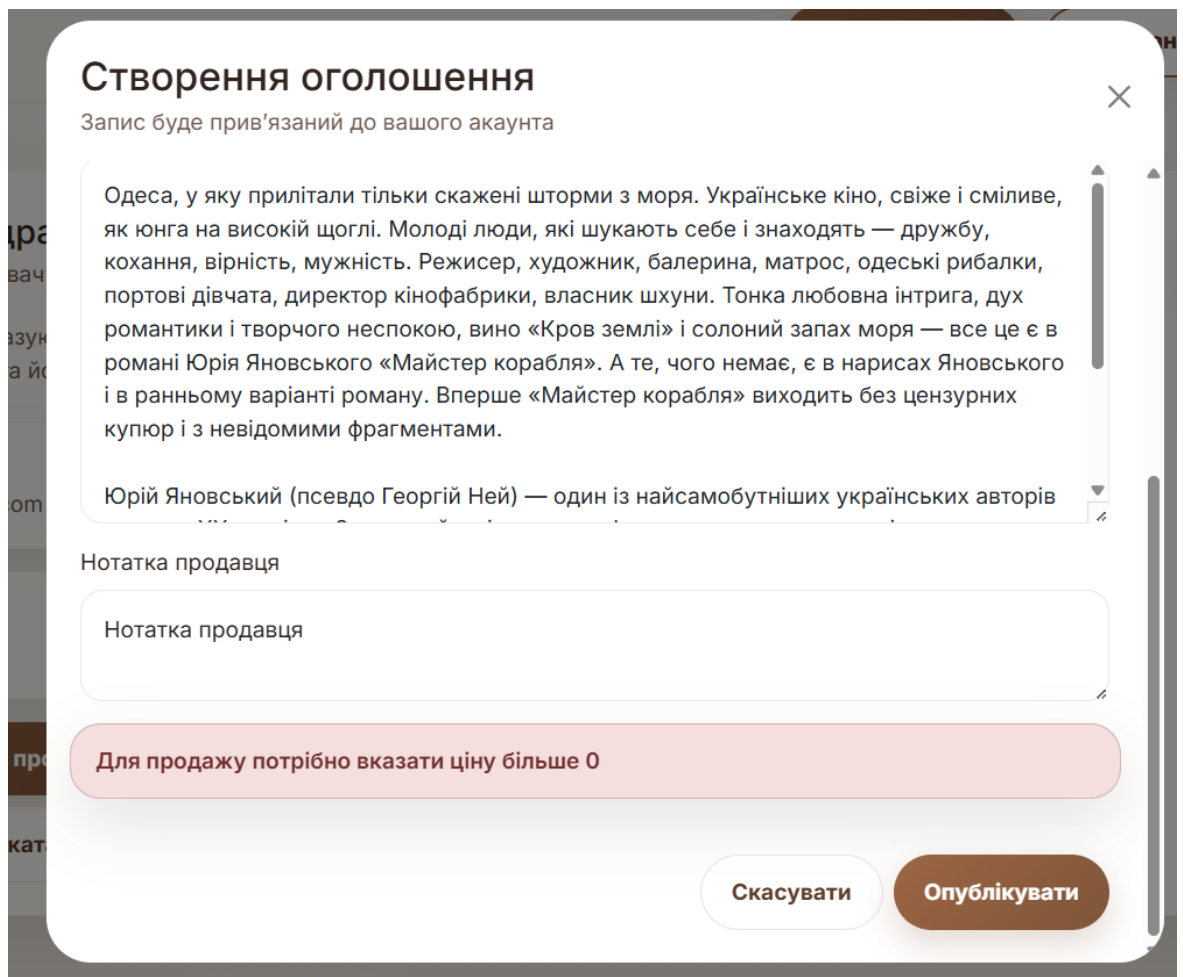


Рисунок 4.10 — Тестування обробки помилок

Результати тестування показали, що основні функціональні можливості мобільного вебзастосунку «BookShelf» працюють відповідно до визначених вимог. Користувач може зареєструватися, увійти в систему, переглядати каталог, шукати й фільтрувати оголошення, створювати власні записи, редагувати та видаляти їх, а також зв'язуватися з продавцем через email.

4.3 Аналіз виявлених недоліків та напрями подальшого розвитку застосунку

Під час тестування мобільного вебзастосунку «BookShelf» було встановлено, що основні функції системи працюють коректно. Водночас тестування також дозволило визначити низку обмежень, які не є критичними для MVP-версії, але можуть бути враховані під час подальшого розвитку застосунку.

Першим обмеженням є відсутність повноцінної системи внутрішніх повідомлень між користувачами. У поточній версії зв'язок із продавцем реалізовано через email. Це спрощує розробку і зменшує складність системи, однак у майбутньому доцільно додати внутрішній чат, щоб користувачі могли обговорювати деталі продажу або обміну без переходу до стороннього поштового клієнта.

Другим обмеженням є відсутність адміністративної панелі. У поточній версії система орієнтована на звичайних користувачів, які створюють і переглядають оголошення. Однак для повноцінної експлуатації застосунку в реальному середовищі доцільно передбачити роль адміністратора. Адміністратор міг би переглядати всі оголошення, блокувати некоректні записи, видаляти спам, перевіряти скарги користувачів і контролювати якість контенту.

Третім напрямом покращення є розширення фільтрації. У поточній версії передбачено фільтрацію за містом і типом оголошення. У майбутньому можна додати фільтри за жанром, автором, станом книги, діапазоном ціни або датою додавання. Це зробить пошук зручнішим, особливо якщо кількість оголошень у системі збільшиться.

Четвертим недоліком є обмежена робота із зображеннями. Якщо користувачі додають посилання на зображення або зображення мають різні розміри, це може впливати на зовнішній вигляд карток оголошень. У майбутньому варто реалізувати завантаження зображень безпосередньо на сервер або в хмарне сховище, а також автоматичне обрізання й оптимізацію зображень.

П'ятим напрямом розвитку є покращення адаптивності та доступності інтерфейсу. Хоча застосунок розроблений як мобільний вебзастосунок, у майбутньому варто додатково перевірити контрастність кольорів, доступність кнопок, зручність навігації з клавіатури та наявність альтернативного тексту для зображень.

Шостим напрямом розвитку є покращення безпеки. У поточній версії

передбачено базові заходи: хешування паролів, перевірку введених даних, контроль доступу до власних оголошень і використання параметризованих SQL-запитів. У майбутньому можна додати повноцінну систему токенів або сесій, відновлення пароля, підтвердження email, обмеження кількості спроб входу та журналювання важливих дій користувачів.

Сьомим напрямом розвитку є додавання статусів оголошень. Наприклад, оголошення може мати статус «активне», «зарезервоване», «продано» або «обміняно». Це допоможе користувачам краще розуміти, чи актуальна пропозиція, а власникам оголошень — не видаляти записи одразу після завершення угоди.

Восьмим напрямом розвитку є можливість додавання відгуків або рейтингу користувачів. Оскільки система передбачає взаємодію між людьми, механізм відгуків може підвищити довіру до продавців і покупців. Однак таку функцію доцільно реалізовувати після базової стабілізації системи, оскільки вона потребує додаткової логіки модерації та захисту від зловживань.

Виявлені обмеження не заважають роботі основних функцій застосунку, але показують, у яких напрямках систему можна розвивати надалі. Для дипломного проєкту поточна MVP-версія є достатньою, оскільки вона реалізує основну мету: створення інформаційної системи для продажу та обміну книг у форматі мобільного вебзастосунку.

№	Обмеження поточної версії	Можливий напрям розвитку
1	Зв'язок із продавцем реалізовано через email	Додати внутрішній чат між користувачами
2	Відсутня адміністративна панель	Додати роль адміністратора та модерацію оголошень
3	Фільтрація обмежена базовими параметрами	Додати фільтри за автором, жанром, станом і ціною
4	Обмежена робота із зображеннями	Реалізувати завантаження, обрізання й оптимізацію фото
5	Немає статусів оголошень	Додати статуси “активне”, “продано”, “обміняно”
6	Немає системи відгуків	Додати рейтинг або відгуки користувачів

7	Базова система безпеки	Додати токени/сесії, підтвердження email і відновлення пароля
8	Обмежена аналітика роботи системи	Додати журнал дій і статистику оголошень

Таблиця 4.2 — Виявлені обмеження та напрями подальшого розвитку

Висновки до розділу 4

У четвертому розділі було проведено тестування мобільного вебзастосунку «BookShelf». Тестування охопило основні функціональні можливості системи: запуск застосунку, реєстрацію користувача, вхід у систему, перегляд каталогу книг, пошук, фільтрацію, створення, редагування та видалення оголошень, перегляд профілю, зв'язок із продавцем і вихід із системи.

Під час тестування було перевірено не лише позитивні сценарії, а й ситуації з некоректними даними: порожні поля, неправильний пароль, спроби виконати дії без належних прав. Такий підхід дозволив оцінити стабільність роботи застосунку та правильність обробки помилок.

Результати тестування показали, що основні функції застосунку працюють відповідно до визначених вимог. Користувач може виконувати базові дії, необхідні для продажу або обміну книг: створювати обліковий запис, переглядати каталог, знаходити потрібні оголошення, додавати власні книги та керувати ними.

Також було визначено напрями подальшого розвитку системи. До них належать додавання внутрішнього чату, адміністративної панелі, розширеної фільтрації, статусів оголошень, відгуків користувачів, покращеної роботи із зображеннями та додаткових механізмів безпеки.

Отже, тестування підтвердило працездатність мобільного вебзастосунку «BookShelf» і показало, що розроблена система може бути використана як основа для подальшого розвитку інформаційної системи продажу та обміну книг.

ЗАГАЛЬНІ ВИСНОВКИ

У кваліфікаційній роботі виконано проєктування та розробку інформаційної системи продажу та обміну книг у форматі мобільного вебзастосунку «BookShelf». За результатами виконаної роботи можна зробити такі висновки.

Проведений аналіз предметної області підтвердив актуальність теми. Книжковий ринок України зростає: у 2024 році загальний наклад видань збільшився на 21%, а рівень інтернет-проникнення досяг 89,6% населення. Існуючі рішення — загальні маркетплейси, онлайн-книгарні та буккросингові сервіси — не поєднують у собі книжкову спеціалізацію, підтримку продажу й обміну, локальний пошук та мобільний інтерфейс. Це обґрунтувало доцільність розробки BookShelf.

Визначено функціональні та нефункціональні вимоги до системи. Сформульовано 13 функціональних вимог, зокрема реєстрацію, авторизацію, перегляд каталогу, пошук, фільтрацію, створення, редагування та видалення оголошень, перегляд профілю і зв'язок із продавцем. Нефункціональні вимоги охоплюють адаптивність, безпеку, надійність, продуктивність і підтримуваність.

Спроектовано архітектуру застосунку за клієнт-серверним принципом з поділом на рівень інтерфейсу, рівень API, рівень бізнес-логіки та рівень даних. Спроектовано базу даних із двома основними таблицями — `users` і `listings` — пов'язаними через зовнішній ключ. Визначено дев'ять REST API-маршрутів для всіх операцій системи.

Реалізовано повнофункціональний мобільний вебзастосунок із використанням стеку HTML, CSS, Bootstrap 5, JavaScript на клієнтській стороні та Node.js, Express.js, MySQL на серверній. Клієнтська логіка зосереджена у файлі `app.js`, серверна організована модульно через окремі роутери `authRoutes.js` і `listingRoutes.js`. Реалізовано хешування паролів за допомогою `bcrypt`, параметризовані SQL-запити та перевірку власника перед редагуванням або видаленням оголошення.

Проведено ручне функціональне тестування за п'ятьма напрямками: функціональне тестування, тестування інтерфейсу, адаптивності, API та обробки помилок. Результати підтвердили, що всі ключові сценарії системи — реєстрація, вхід, перегляд каталогу, пошук, фільтрація, CRUD-операції з оголошеннями та email-зв'язок — працюють відповідно до визначених вимог.

Визначено обмеження поточної MVP-версії та напрями подальшого розвитку: додавання внутрішнього чату, адміністративної панелі, розширених фільтрів, завантаження зображень на сервер, системи статусів оголошень, механізму відгуків і повноцінної системи токенів.

Практичне значення роботи полягає у створенні функціонального вебзастосунку, який може бути використаний для організації продажу та обміну книг між користувачами. Застосунок реалізує основний сценарій взаємодії — від реєстрації до зв'язку з продавцем — і може слугувати основою для подальшого розширення та розгортання в реальному середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буккросинг. Чернігівська обласна універсальна наукова бібліотека імені Софії та Олександра Русових. URL: <https://libkor.com.ua/page/153-bukkrosing> (дата звернення: 10.05.2026).
2. Оголошення olx.ua: сервіс оголошень України – купівля/продаж бу та нових товарів, різноманітні послуги на сайті olx.ua. Оголошення OLX.ua: сервіс оголошень України – купівля/продаж бу та нових товарів, різноманітні послуги на сайті OLX.ua. URL: <https://www.olx.ua/uk/hobbi-otdyh-i-sport/knigi-zhurnaly/?srsltid=AfmBOoqQSuWDDVZFLtzaiv2kz9nrkZBSWh6Rm3a1PRjZrSBS-tYPqAr9> (дата звернення: 10.05.2026).
3. У Telegramі з'явився сервіс із безкоштовного обміну книжками. Читомо | Культура читання і мистецтво книговидавання | Усе про книги | Читомо - портал про культуру читання і мистецтво книговидавання. URL: <https://chytomo.com/u-telegrami-z-iavyvsia-servis-iz-bezkoshtovnoho-obminu-knyzhkamuy/> (дата звернення: 10.05.2026).
4. Український інститут книги. URL: <https://ubi.org.ua/uk/activity/doslidzhennya/doslidzhennya-2025> (дата звернення: 10.05.2026).
5. Що таке вебдодатки та як вони функціонують? | Asabix. Asabix. URL: <https://asabix.com.ua/what-are-web-applications-and-how-do-they-function/> (дата звернення: 10.05.2026).
6. AbeBooks | shop for books, art & collectibles. AbeBooks. URL: <https://www.abebooks.com/> (date of access: 10.05.2026).
7. About the unified modeling language specification version 2.5.1. Object Management Group. URL: <https://www.omg.org/spec/UML/2.5.1/About-UML> (date of access: 10.05.2026).
8. Authorization - OWASP cheat sheet series. Introduction - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.ht

- ml (date of access: 10.05.2026).
9. Breakpoints. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: <https://getbootstrap.com/docs/5.0/layout/breakpoints> (date of access: 10.05.2026).
 10. Cross-Origin Resource Sharing (CORS) - HTTP | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS> (date of access: 10.05.2026).
 11. Digital 2026: ukraine – datareportal – global digital insights. DataReportal – Global Digital Insights. URL: <https://datareportal.com/reports/digital-2026-ukraine> (date of access: 10.05.2026).
 12. Express - Node.js web application framework. Express - Node.js web application framework. URL: <https://expressjs.com> (date of access: 10.05.2026).
 13. Express cors middleware. Express - Node.js web application framework. URL: <https://expressjs.com/en/resources/middleware/cors.html> (date of access: 10.05.2026).
 14. Express routing. Express - Node.js web application framework. URL: <https://expressjs.com/en/guide/routing.html> (date of access: 10.05.2026).
 15. Express/Node introduction - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction (date of access: 10.05.2026).
 16. Fetch API - Web APIs | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (date of access: 10.05.2026).
 17. Grid system. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: <https://getbootstrap.com/docs/5.0/layout/grid/> (date of access: 10.05.2026).
 18. HTTP: hypertext transfer protocol | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP> (date of access: 10.05.2026).

19. HTTP request methods - HTTP | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods> (date of access: 10.05.2026).
20. HTTP response status codes - HTTP | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status> (date of access: 10.05.2026).
21. IEEE standards association. IEEE Standards Association. URL: <https://standards.ieee.org/ieee/29148/12262> (date of access: 10.05.2026).
22. Input validation - OWASP cheat sheet series. Introduction - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html (date of access: 10.05.2026).
23. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models [Електронний ресурс]. First edition. 2011. URL: <https://cdn.standards.iteh.ai/samples/35733/2ca18b477b7845a5b8cae39d6de0c098/ISO-IEC-25010-2011.pdf> (date of access: 10.05.2026).
24. Jonker A., Gomstyn A. What is requirements management? | IBM. IBM. URL: <https://www.ibm.com/think/topics/what-is-requirements-management> (date of access: 10.05.2026).
25. Least privilege principle | OWASP foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-community/controls/Least_Privilege_Principle (date of access: 10.05.2026).
26. Media query fundamentals - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Media_queries (date of access: 10.05.2026).
27. Mobile First дизайн: що це і як правильно реалізувати мобільну версію сайту?. ІТ-компанія з розробки цифрових продуктів в Україні – Brander.

- URL: <https://brander.ua/blog/mobile-first-design-shcho-tse-i-yak-yoho-pravylyno-realizuvaty> (дата звернення: 10.05.2026).
28. Mobile-first indexing best practices | google search central | documentation | google for developers. Google for Developers. URL: <https://developers.google.com/search/docs/crawling-indexing/mobile/mobile-sites-mobile-first-indexing> (date of access: 10.05.2026).
29. Mobile web application best practices cards. W3C. URL: <https://www.w3.org/2010/09/MWABP/> (date of access: 10.05.2026).
30. MySQL :: MySQL 8.4 Reference Manual :: 15.1.20.5 FOREIGN KEY Constraints. fw_error_www. URL: <https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html> (date of access: 10.05.2026).
31. Nielsen J. 10 usability heuristics for user interface design. Nielsen Norman Group. URL: <https://www.nngroup.com/articles/ten-usability-heuristics> (date of access: 10.05.2026).
32. Node.js – about node.js®. Node.js – Run JavaScript Everywhere. URL: <https://nodejs.org/en/about> (date of access: 10.05.2026).
33. Overview of HTTP - HTTP | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview> (date of access: 10.05.2026).
34. Password storage - OWASP cheat sheet series. Introduction - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (date of access: 10.05.2026).
35. Responsive web design - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design (date of access: 10.05.2026).
36. SQL injection prevention - OWASP cheat sheet series. Introduction - OWASP Cheat Sheet Series. URL:

- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (date of access: 10.05.2026).
37. Using the Fetch API - Web APIs | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (date of access: 10.05.2026).
38. Validate all inputs - OWASP developer guide. Introduction - OWASP Developer Guide. URL: <https://devguide.owasp.org/en/04-design/02-web-app-checklist/05-validate-inputs> (date of access: 10.05.2026).
39. Web API design best practices - azure architecture center. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design> (date of access: 10.05.2026).
40. Web content accessibility guidelines (WCAG) 2.2. W3C. URL: <https://www.w3.org/TR/WCAG22> (date of access: 10.05.2026).
41. Web forms - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms (date of access: 10.05.2026).
42. WSTG - latest | OWASP foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/README (date of access: 10.05.2026).

ДОДАТКИ

Додаток А

Серверна частина (server.js) — точка входу: підключення middleware, роутерів, запуск.

```
const express = require('express');
const cors = require('cors');
require('dotenv').config();

const db = require('./config/db');
const authRoutes = require('./routes/authRoutes');
const listingRoutes = require('./routes/listingRoutes');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(express.json());

db.connect((err) => {
  if (err) {
    console.error('Помилка підключення до MySQL:', err);
    return;
  }

  console.log('Підключено до MySQL');
});

app.get('/', (req, res) => {
  res.send('BookShelf API працює');
});

app.use('/api', authRoutes);
app.use('/api', listingRoutes);

app.listen(PORT, () => {
  console.log(`Сервер запущено на http://localhost:${PORT}`);
});
```

Додаток Б

Маршрути авторизації (routes/authRoutes.js) — реєстрація, вхід, оновлення профілю.

```
router.post('/register', async (req, res) => {
  try {
    const { name, email, password, city } = req.body;

    if (!name || !email || !password || !city) {
      return res.status(400).json({
        message: 'Усі поля обов'язкові',
      });
    }

    if (password.length < 6) {
      return res.status(400).json({
        message: 'Пароль має бути мінімум 6 символів',
      });
    }

    const checkUserQuery = 'SELECT id FROM users WHERE email = ?';

    db.query(checkUserQuery, [email], async (checkErr, checkResult)
=> {
      if (checkErr) {
        console.error('Помилка перевірки користувача:', checkErr);
        return res.status(500).json({
          message: 'Помилка сервера при перевірці email',
        });
      }

      if (checkResult.length > 0) {
        return res.status(409).json({
          message: 'Користувач з таким email вже існує',
        });
      }

      const hashedPassword = await bcrypt.hash(password, 10);

      const insertUserQuery = `
        INSERT INTO users (name, email, password, city)
        VALUES (?, ?, ?, ?)
      `;

      db.query(
        insertUserQuery,
        [name, email, hashedPassword, city],
        (insertErr, insertResult) => {
          if (insertErr) {
            console.error('Помилка створення користувача:',
```

```

insertErr);
    return res.status(500).json({
      message: 'Помилка сервера при створенні користувача',
    });
  }

  return res.status(201).json({
    message: 'Користувача зареєстровано успішно',
    userId: insertResult.insertId,
  });
},
);
});
} catch (error) {
  console.error('Загальна помилка:', error);
  return res.status(500).json({
    message: 'Внутрішня помилка сервера',
  });
}
});

router.post('/login', (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({
      message: 'Email і пароль обов’язкові',
    });
  }

  const query = 'SELECT * FROM users WHERE email = ?';

  db.query(query, [email], async (err, results) => {
    if (err) {
      console.error('Помилка логіну:', err);
      return res.status(500).json({
        message: 'Помилка сервера',
      });
    }

    if (results.length === 0) {
      return res.status(401).json({
        message: 'Невірний email або пароль',
      });
    }

    const user = results[0];

    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {

```

```

        return res.status(401).json({
            message: 'Невірний email або пароль',
        });
    }

    return res.json({
        message: 'Успішний вхід',
        user: {
            id: user.id,
            name: user.name,
            email: user.email,
            city: user.city,
        },
    });
});
});
});

```

Додаток В

Підключення до бази даних (config/db.js)

```

const mysql = require('mysql2');

const db = mysql.createConnection({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME,
});

module.exports = db;

```

Додаток Г

Клієнтська логіка (assets/js/app.js)

```

//Завантаження оголошень книг
async function loadBooks(filters = {}) {
    const params = new URLSearchParams();

    if (filters.title) {
        params.set('title', filters.title);
    }

    if (filters.city && filters.city !== 'all') {
        params.set('city', filters.city);
    }

    if (filters.type && filters.type !== 'all') {
        params.set('type', uiTypeToApi(filters.type));
    }
}

```

```

    const queryString = params.toString();
    const url = queryString ? `/api/listings?${queryString}` :
'/api/listings';

    const data = await apiFetch(url);
    const normalized = data.map(normalizeBook);
    state.books = normalized;

    return normalized;
}

```

//Редагування оголошень

```

async function handleListingSubmit(event) {
    event.preventDefault();

    const form = event.currentTarget;
    clearListingFormError(form);

    if (!requireAuth()) return;

    const editId = form.dataset.editId;
    const payload = collectFormData(form);

    try {
        if (editId) {
            await apiFetch(`/api/listings/${editId}`, {
                method: 'PUT',
                body: JSON.stringify({
                    ...payload,
                    user_id: state.currentUser.id,
                }),
            });

            messageBox('Оголошення оновлено');
        } else {
            await apiFetch('/api/listings', {
                method: 'POST',
                body: JSON.stringify({
                    ...payload,
                    user_id: state.currentUser.id,
                }),
            });

            messageBox('Оголошення створено');
        }

        form.reset();
        delete form.dataset.editId;
        clearListingFormError(form);
        syncPriceField(form);
    }
}

```

```

    const modal = bootstrap.Modal.getInstance(qs('#addBookModal'));
    if (modal) modal.hide();

    await refreshPageData();
  } catch (error) {
    showListingFormError(form, error.message);
  }
}

//Перевірка власника оголошення
function isOwner(book) {
  return Boolean(
    state.currentUser && Number(book.userId) ===
    Number(state.currentUser.id),
  );
}

//Видалити оголошення
async function deleteListing(id) {
  if (!requireAuth()) return;
  if (!confirm('Видалити це оголошення?')) return;

  try {
    await apiFetch(`/api/listings/${id}`, {
      method: 'DELETE',
      body: JSON.stringify({
        user_id: state.currentUser.id,
      }),
    });

    messageBox('Оголошення видалено');
    await refreshPageData();
  } catch (error) {
    messageBox(error.message, 'error');
  }
}

```

Додаток Д

SQL-скрипт бази даних (database.sql)

CREATE TABLE для users і listings, CREATE INDEX.

```

DROP DATABASE IF EXISTS book_app;
CREATE DATABASE book_app CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;

```

```

USE book_app;

```

```

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,

```

```

    name VARCHAR(120) NOT NULL,
    email VARCHAR(120) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    city VARCHAR(80) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

CREATE TABLE listings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,

    title VARCHAR(200) NOT NULL,
    description TEXT,
    condition_book ENUM('new', 'excellent', 'good', 'used') NOT NULL
DEFAULT 'good',
    type ENUM('sell', 'exchange') NOT NULL,
    city VARCHAR(80) NOT NULL,

    image_url TEXT NULL,
    price DECIMAL(10,2) NULL,
    seller_note TEXT NULL,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE INDEX idx_listings_city ON listings(city);
CREATE INDEX idx_listings_type ON listings(type);
CREATE INDEX idx_listings_user_id ON listings(user_id);

```

Додаток Е

Робота вебзастосунку з телефона

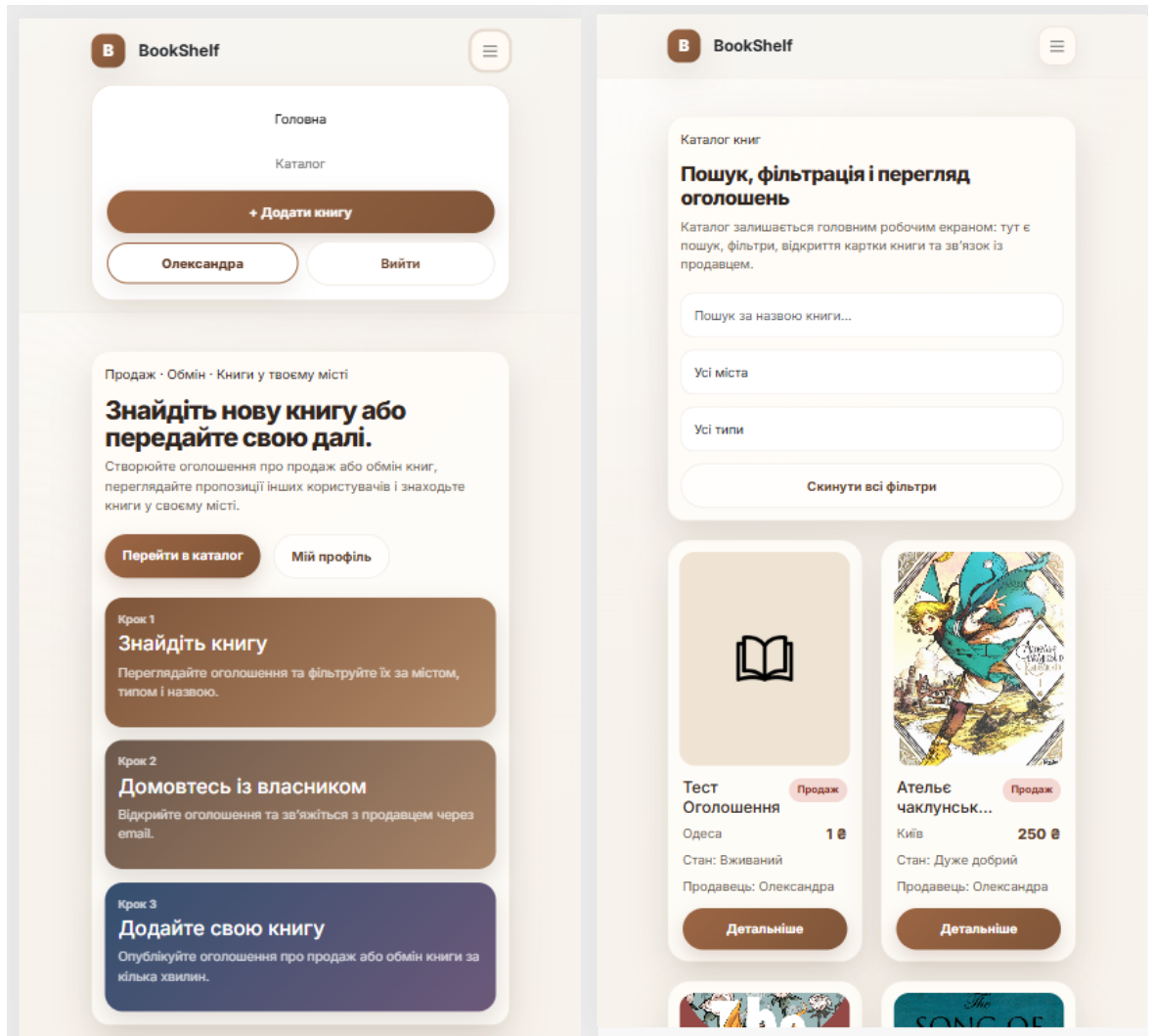


Рисунок Е.1

Рисунок Е.2

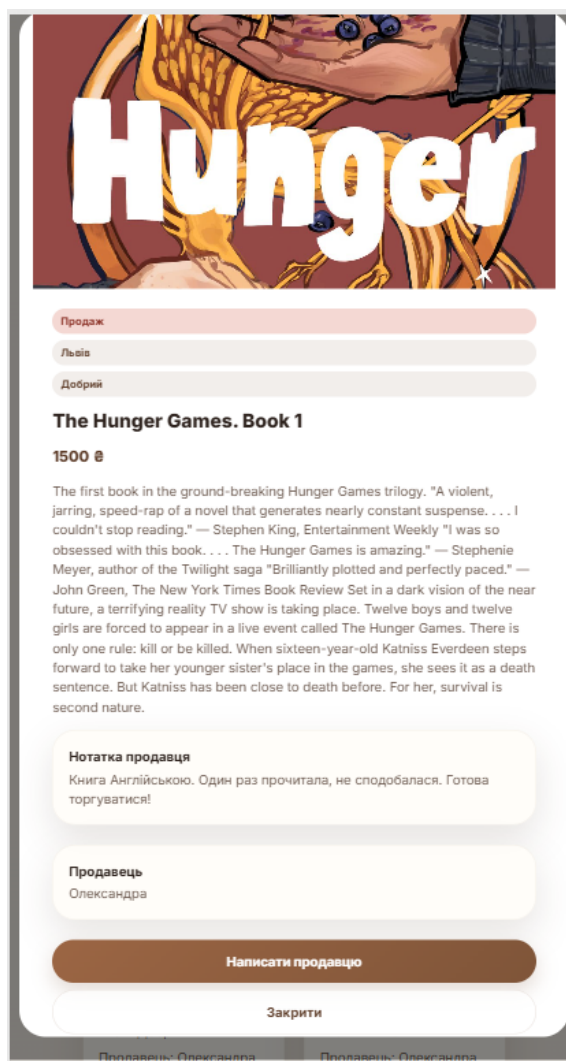


Рисунок Е.3

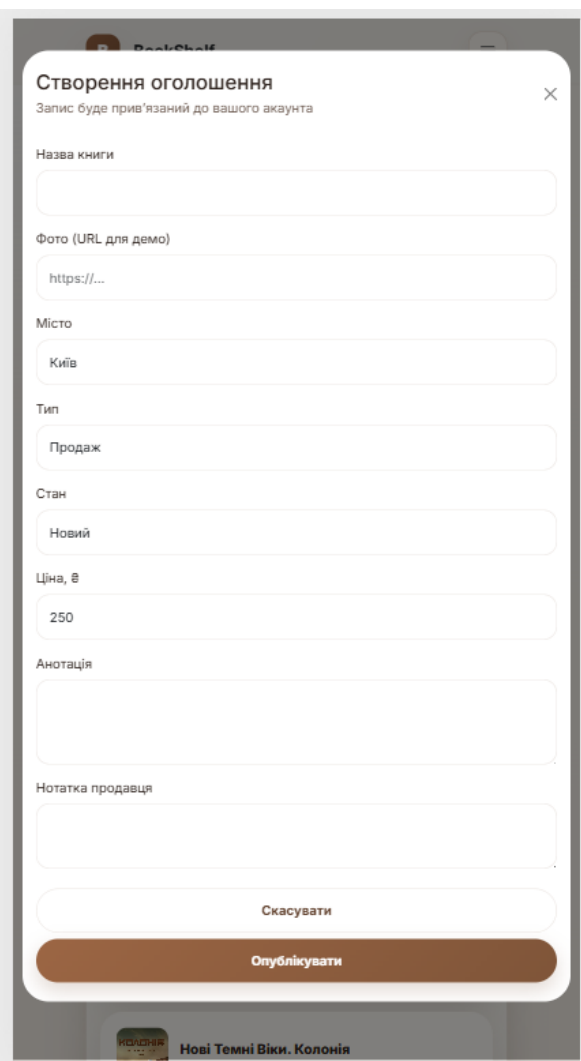


Рисунок Е.4