ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку по обміну друкованими книгами мовою C# та React»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Михайло СКВОРЦОВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Михайло СКВОРЦОВ

Керівник: Ірина ЗАМРІЙ
д.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Скворцову Михайлу Олександровичу

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку по обміну друкованими книгами мовою C# та React»

керівник кваліфікаційної роботи д.т.н., доцент Ірина ЗАМРІЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про сучасні технології розробки веб-застосунків, опис архітектурних підходів для створення багаторівневих систем, технічну документацію з описом використаних фреймворків та бібліотек, таких як Next.js, ASP.NET Web API та SQL Server.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Фундаментальні аспекти розробки веб-застосунку
2. Методологія проектування веб-застосунку.
3. Реалізація та демонстрація функціональності веб-застосунку

4. Верифікація та оцінка ефективності веб-застосунку

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до застосунку.
3. Програмні засоби реалізації.
4. Use Case діаграма.
5. Діаграма діяльності.
6. Діаграма бази даних
7. Мапа сайту.
8. Екранні форми.
9. Відео роботи сайту
10. Апробація результатів дослідження

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Вивчення ІТ-засобів, придатних для реалізації веб-застосунку	14.03.-20.03.2024	
4	Проектування веб-застосунку обміну друкованими книгами	21.03.-10.04.2024	
5	Реалізація функціональності веб-застосунку	11.04-25.04.2024	
6	Верифікація та оцінка ефективності веб-застосунку	26.04-28.05.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Михайло СКВОРЦОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 11 табл., 38 рис., 20 джерел.

Мета роботи – спрощення процесу обміну друкованими книгами.

Об’єкт дослідження – обмін друкованими книгами.

Предмет дослідження – web-застосунок, який спрямований на обмін друкованими книгами.

Короткий зміст роботи: У даній кваліфікаційній роботі проведено аналіз існуючих веб-платформ для обміну друкованими книгами та виявлено основні недоліки та обмеження їхньої функціональності. Досліджено та порівняно ключові технології, які застосовуються у розробці веб-застосунків, з особливим акцентом на використанні React для розробки фронтенду та C# для розробки бекенду. Розроблено архітектуру системи, яка включає декілька рівнів взаємодії користувачів з платформою, в тому числі системи реєстрації, авторизації, пошуку книг та управління профілями. Програмно реалізовані основні модулі системи, які забезпечують користувачам доступ до функцій перегляду, управління книгами та інтерактивні можливості, такі як чати та відгуки.

Сферою застосування додатку є обмін друкованими книгами, що сприяє збереженню ресурсів та підтримці культурного обміну між користувачами різних регіонів.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, C#, РОЗРОБКА, АРХІТЕКТУРА, ІНТЕРФЕЙС КОРИСТУВАЧА

ЗМІСТ

ВСТУП

Розробка веб-застосунків з використанням різних технологій набуває все більшого значення в цифрову епоху, впливаючи на численні галузі та змінюючи спосіб взаємодії з цифровим контентом. Моя робота враховує цю тенденцію, звертаючись до конкретної потреби в платформі, яка полегшує обмін фізичними книгами за допомогою зручного, ефективного та масштабованого веб-застосунку.

Ця тема важлива, оскільки споживачі все частіше обирають екологічні та економічні способи читання книг. Незважаючи на те, що існує більше способів доступу до інформації в Інтернеті, люди все одно люблять читати друковані книги, проте знайти чи отримати необхідну книгу не так і просто. Я хочу вирішити деякі проблеми, з якими стикаються користувачі паперових книги, створивши веб-сайт, де люди зможуть обмінювати непотрібні книги на інші, щоб книги служили довше та були більш екологічними.

Після критичного аналізу сучасних рішень стало очевидним, що хоча існують платформи для купівлі та продажу книг, майданчики для обміну друкованою літературою залишаються недостатньо розвинутими. Більшість наявних сервісів зосереджена на електронних форматах, тоді як багато читачів все ще віддають перевагу фізичним копіям. Мій аналіз публікацій та досліджень у даній галузі підтвердив, що є потреба у спеціалізованому застосунку для обміну друкованими книгами.

Метою кваліфікаційної роботи є спрощення процесу обміну друкованими книгами.

Завданнями дослідження є:

1. Дослідження сучасних тенденцій та потреб у сфері обміну друкованими книгами

2. Аналіз існуючих рішень на ринку, їхніх функціональних можливостей, переваг та недоліків.
3. Проектування архітектури веб-додатку, забезпечення його масштабованості, безпеки.
4. Проектування та розробка бази даних системи обміну друкованими книгами.
5. Проектування інтерфейсу користувача системи обміну друкованими книгами.
6. Розробка та імплементація основних функцій веб-застосунку обміну друкованими книгами.
7. Проведення мануального тестування системи на відповідність вимогам.
8. Проходження апробації на Науково-технічних конференціях.

Об'єкт дослідження - обмін друкованими книгами.

Предмет дослідження - це web-застосунок, який спрямований на обмін друкованими книгами, що буде включати функціонал для пошуку, додавання та обміну книгами, а також можливість залишати відгуки та рейтинги книг.

Однією з особливостей цього застосунку є можливість користувачів створювати свої власні списки книг, які вони хочуть отримати, і використовувати чат для прямого спілкування між собою, що сприяє ефективнішому та цілеспрямованій взаємодії.

Ці характеристики перетворюють веб-застосунок з платформи для обміну на місце спілкування, де користувачі можуть ділитися враженнями про книги та будувати спільноту. Цей метод важливий для бібліотек, книгарень і приватних осіб, оскільки спонукає до читання та розширює доступ до різних книжкових ресурсів.

Інноваційність мого проекту полягає в комплексному підході до розробки веб-застосунку для обміну книгами, який поєднує кілька основних функцій для покращення продуктивності та задоволення потреб користувачів. Специфічним

нововведенням є впровадження системи перегляду авторів і книг, яка не лише надає інформацію, а й дозволяє користувачам краще вивчити кожен твір та його автора. Це містить докладні описи, історичний контекст, зв'язки між творами та інформацію про авторів, що може допомогти вибрати кращу книгу для читання та обміну.

Іншою новітньою функцією є можливість створення списку улюблених книг у застосунку, яка дозволяє користувачам відзначати книги, які вони хотіли б мати в майбутньому, що полегшує більш спрямований обмін.

Чат у моєму веб-застосунку сприяє обговоренню деталей обміну книгами та формуванню активної спільноти літературних ентузіастів. Він перетворює платформу з простого сервісу обміну в багатофункціональний простір, де учасники можуть ділитися враженнями від прочитаних книг, обмінюватися ідеями та збагачувати свій культурний досвід, що робить застосунок важливим ресурсом для культурного розвитку та спілкування.

Практична значимість роботи проявляється в тому, що веб-застосунок може бути використаний для підтримки спільноти читачів і бібліотек, які прагнуть зберегти і розширити доступ до фізичних книг через організацію ефективного обміну без зайвих витрат, пов’язаних із фінансовими операціями.

Апробація результатів відбулася під час участі в конференціях, де я мав можливість демонструвати функціональні можливості веб-застосунку і отримувати зворотній зв'язок від інших розробників та користувачів. Ці виступи сприяли подальшому розвитку проекту. [1]

Дослідження у рамках цієї дипломної роботи спрямоване на аналіз сучасних технологій розробки web-застосунків, вивчення принципів роботи з базами даних, а також оптимізацію взаємодії між бекендом і фронтендом.

1 ФУНДАМЕНТАЛЬНІ АСПЕКТИ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ

1.1 Аналіз предметної галузі та формулювання задач дослідження

В контексті розробки веб-застосунку для обміну друкованими книгами, важливо проаналізувати предметну галузь для розуміння тенденцій та потреб на ринку. Це включає всі аспекти, пов'язані з процесом обміну книгами через веб-платформу.

Перш за все, це огляд самого ринку обміну книгами. Цей ринок і книжкова індустрія загалом продемонстрували стійкість у відповідь на технологічний прогрес і зміну споживчих уподобань. Відповідно до рис.1.1, у 2023 році світовий книжковий ринок, оцінювався в 144,67 мільярда доларів США, і очікується, що з 2024 по 2030 рік він зростатиме на 1,8% у річному обсязі.

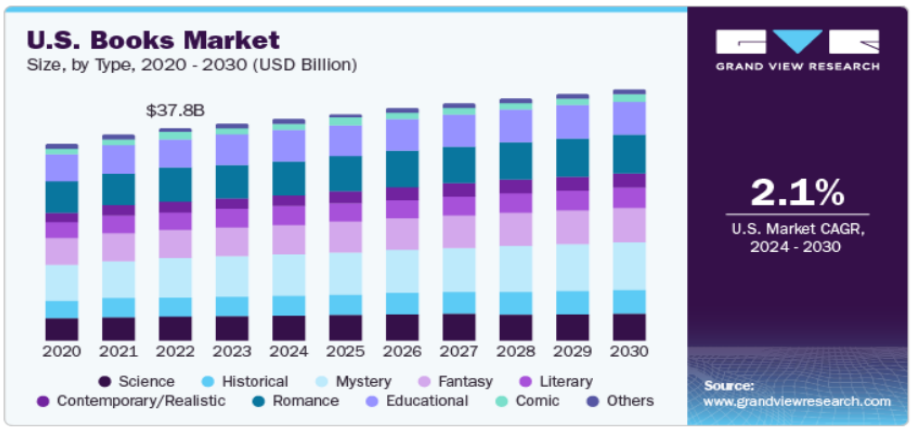


Рис. 1.1 Ринок книг 2020 - 2030 рр

Незважаючи на зростання цифрових медіа, ринок фізичних книг залишається сильним. У 2023 році на друковані книжки припадало понад 78% світових доходів від книжкової продукції. Одним із ключових викликів на ринку є зростання цін на друковані та електронні книги, через що споживачі зараз шукають альтернативні варіанти, а не купують нові примірники. Таким чином, високі ціни на книги призводять до того, що споживачі знаходять дешевші альтернативи, наприклад,

купують вживані книги або завантажують безкоштовні електронні книги. В зв'язку з цим популярність платформ для обміну книгами зростає та має своїх поціновувачів. Оскільки фізичні книги мають свої переваги, такі як легкість перегляду та менше навантаження на очі порівняно з цифровими форматами.

На рис. 1.2 відображені канали розповсюдження книг. Відповідно до рисунка, домінуючим каналом розповсюдження були місцеві книжкові магазини, з часткою глобального доходу понад 50% у 2023 році.

Рис. 1.1 Канали розповсюдження книг

Проте очікується, що онлайн-канал зросте на 2,9% CAGR з 2024 по 2030 рік. Тому розробка web-застосунку для обміну книгами є доволі актуальною [2].

Важливим аспектом також є поведінка користувачів. Щоб зрозуміти поведінку споживачів на платформах онлайн-книгообміну, вкрай важливо розглянути вплив електронного сарафанного спілкування (eWOM) і взаємодії в соціальних мережах. Дослідження показали, що eWOM значно впливає на поведінку споживачів, надаючи користувачам платформу для обміну думками та досвідом. Споживачі все більше покладаються на онлайн-огляди та соціальні мережі, щоб приймати обґрунтовані рішення щодо продуктів, зокрема книг. Така поведінка зумовлена довірою та незалежністю відгуків від інших користувачів, які часто вважаються більш достовірними, ніж традиційна реклама чи інформація, надана продавцем. Тому коли споживачам потрібна інформація про продукт або

послугу, вони в кінцевому підсумку звертаються до онлайн-медіа з двох причин. По-перше, вони можуть отримати інформацію швидше, оскільки не потрібно чекати, поки хтось інший - висловить свою думку про те, що вони хочуть споживати. По-друге, вони можуть використовувати онлайн-медіа для підтвердження отриманої інформації. Онлайновий характер цих взаємодій забезпечує широке охоплення, тобто на споживачів впливає не лише місцева чи особиста взаємодія, але й глобальна онлайн-спільнота. Перехід до цифрових платформ змінив те, як споживачі збирають інформацію, порівнюють продукти та, зрештою, приймають рішення про покупку [3]. В табл. 1.1 відображена різниця між WOM та eWOM

Таблиця1.1

Різниця між «сарафанним маркетингом» (WOM) та «електронним сарафанним маркетингом»(eWOM) [2]

	WOM	eWOM
Достовірність	Одержувач інформації знає комунікатора (позитивний вплив на довіру)	Анонімність між комунікатором і одержувачем інформації (негативний вплив на довіру)
Конфіденційність	Розмова є приватною, міжособистісною (через діалоги) і ведеться в режимі реального часу	Спільна інформація не є конфіденційною, і, оскільки вона записана, іноді може переглядатися будь-ким і в будь-який час
Швидкість	Повідомлення поширюються повільно. Користувачі повинні бути присутні під час передачі інформації	Повідомлення передаються між користувачами швидше і через Інтернет можуть бути передані в будь-який час
Доступність	Менш доступний	Легкодоступний

Як результат важливо реалізувати на платформі можливість лишати відгуки,оцінки та вести комунікацію між користувачами.

Крім того, еволюція Інтернету з часом призвела до змін у поведінці споживачів. Спочатку Інтернет служив головним чином як нова платформа для обміну інформацією, але згодом він розвинувся, щоб сприяти більш різноманітним взаємодіям, а тепер має на меті надавати більш персоналізовану та контекстуально відповідну інформацію. Цей прогрес зробив онлайн-платформи, включно з сайтами книгообміну, невід'ємною частиною того, як споживачі відкривають, оцінюють і приймають рішення про свій вибір книг. І відгуки чи то позитивні чи негативні грають вирішальну роль. Як правило, позитивні - викликають емоційну довіру, збільшують довіру до продукту та мають сильний переконливий ефект. Негативні ж коментарі навпаки можуть зменшити формування емоційної довіри та перешкодити намірам споживачів купувати. Крім того, споживачі зазвичай вважають, що негативна інформація є більш цінною, ніж позитивна, коли приймають рішення. Наприклад, оцінка в одну зірку, як правило, має більший вплив на купівельні тенденції споживачів, ніж оцінка в п'ять зірок. Отже, зосередженість на ступені уваги до конкретного змісту коментарів може спонукати споживачів приймати різні рішення про покупку.

Розуміння цієї динаміки може допомогти пристосувати маркетингові та операційні стратегії для книгообміну в Інтернеті, щоб краще задовольняти потреби споживачів і посилювати залучення користувачів [4].

Також доцільно буде ознайомитись з правовим аспектом. Дослідження юридичних аспектів книгообміну в Інтернеті стосується кількох важливих сфер, зокрема питань авторського права та захисту даних. Інтернет-книгообмінники повинні керуватися законами про авторські права, які захищають права авторів і видавців. Це включає забезпечення того, щоб цифрові копії книг не розповсюджувалися незаконно. Платформи, що сприяють обміну книгами, повинні забезпечувати дотримання місцевих і міжнародних норм авторського права, щоб уникнути юридичних наслідків. Оскільки платформа, що розробляється в межах

дипломної роботи не включає обмін електронними примірниками детально це питання не досліджувалось. Проте інше питання – захист даних та інформації є важливим для розробленого проекту. Оскільки, загальний регламент захисту даних (GDPR) в ЄС значно впливає на те, як онлайн-платформи мають обробляти персональні дані. Згідно з GDPR, платформи зобов'язані отримати чітку згоду користувачів перед збором, зберіганням або використанням їхніх особистих даних. Це включає в себе будь-які особисті дані, якими обмінюються під час операцій з книгами або спілкування на платформі. Дані повинні оброблятися прозоро, використовуватися лише для визначених законних цілей і бути захищеними від несанкціонованого доступу [5].

При розробці web-застосунку аналіз усіх аспектів предметної галузі відіграв важливу роль та сприяв коректному розподілу функцій та даних.

1.2 Критичний огляд програмних продуктів для вирішення аналогічних задач

Ринок платформ для обміну книгами пропонує користувачам зручні способи обміну, продажу та покупки книг. Вивчаючи платформи для обміну книгами, їх функціональності та популярності, зосередимось на особливостях, які забезпечують високу залученість користувачів та зручність обміну, а також розглянемо платформи котрі припинили свою діяльність з ряду причин. Однією з популярних раніше платформ, яка попри це зазнала краху була - BookMooch.

Особливістю цієї платформи була можливість отримання балів за додавання книг та обмін цих балів на інші книжки. Користувачі повинні були надіслати принаймні 1 книгу на кожні 2 отримані. Якщо ж це співвідношення не підтримується, то заблоковується можливість отримувати нові книги, навіть якщо є бали, доки коефіцієнт не покращиться. З однієї сторони це сприяло активності обміну та рециклінгу книг, проте з іншої зменшувало кількість користувачів платформи , оскільки деякі користувачі вважали систему балів складною або

неінтуїтивною, особливо новачки. На сайті були доступні функції соціальної взаємодії, такі як форуми та обговорення, які допомагали користувачам обмінюватися досвідом і порадами, створюючи сильне відчуття спільноти. Проте вагомим недоліком цієї платформи був обмежений вибір книг та незручний та інтуїтивно незрозумілий інтерфейс, який не оновлювався досить тривалий час. На сайті була присутня спроба створити фільтрацію, проте реалізована вона, як список жанрів чи інших критерій фільтрування. Саме зовнішній вигляд сайту зменшував кількість користувачів, проте це було не єдиною причиною краху [6]. BookMooch критикували також за те, що він не є інноваційним і не реагує на зміну потреб і вподобань користувачів, що могло призвести до зниження його популярності. І через певний час сервіс зіткнувся з конкуренцією з боку інших веб-сайтів обміну книгами.

Ці веб-сайти пропонували подібні послуги та функції, а деякі з них мали більшу базу користувачів. А у BookMooch була спеціальна база користувачів, але йому було важко залучити нових і розширити охоплення за межі нішової спільноти. Мабуть, однією з критичних причинн стало те, що робота веб-сайту вимагає постійного обслуговування та підтримки, і веб-сайт став надто дорогим для обслуговування. Він покладався на пожертви користувачів, але цих пожертв було недостатньо, щоб покрити витрати. Загалом невдача BookMooch, ймовірно, була спричинена поєднанням цих факторів. Однак веб-сайт був популярним протягом багатьох років і надавав цінні послуги книголюбам у всьому світі, тому його аналіз є доволі важливим для реалізації нової платформи по обміну друкованими книгами [7].

Було проаналізовано також платформу GoodReads, хоча обмін книгами не є основною функцією, GoodReads дозволяє користувачам організовувати обмін книгами через спеціальні групи та форуми [8]. Goodreads є одним із найпопулярніших сайтів в цій сфері із 90 мільйонами зареєстрованих користувачів станом на 2019 рік [9]. Користувачі на платформі можуть створювати власні профілі, вести список прочитаних книг, писати рецензії та отримувати

рекомендації. Під кожною книжкою на сайті є розділ під назвою «спільнота», до якої мають доступ усі користувачі Goodreads. Цей розділ містить графік, який показує п'ятизірковий рейтинг спільноти для книги, а також рецензії на книгу від інших учасників спільноти. Учасники спільноти Goodreads, можна розділити на дві групи. Перша група - це група звичайних учасників , які прийшли дізнатися про книгу. Друга група - це група рецензентів, які радять необізнаним учасникам, чи варто читати книгу, оцінюючи її за п'ятибальною шкалою. Такий розподіл надає рецендентам сильний вплив на користувачів, які тільки бажають прочитати книгу, який не завжди може бути справедливим. Неспроможність Goodreads обмежити роль рецендентів - це велика проблема усієї системи [10]. Однією з переваг платформи є співпраця з онлайн-магазинами книг, що дозволяє користувачам легко купувати книги через посилання. Проте разом з цією перевагою поруч стоїть і недолік, оскільки сильна інтеграція з комерційними книжковими магазинами може створити враження комерціалізації, що може відштовхнути деяких користувачів. Проаналізувавши трафік сайту goodreads.com отримуємо наступні результати:

- Від лютого до квітня 2024 року сайт мав 325.8 мільйонів відвідувань, що на 5.97% менше в порівнянні з попереднім періодом.
- Середньомісячна кількість відвідувань становить 108.6 мільйонів.
- Середній час перебування на сайті становить 4 хвилини 21 секунду.
- 51.37% відвідувачів залишають сайт після перегляду лише однієї сторінки.

Відповідно, можна підсумувати, що сайт компанії має доволі велику кількість відвідувачів, проте більша половина з них не користується послугами платформи [11].

Наступним продуктом для вирішення поставлених задач є PaperBackSwap. Платформа має великий вибір книг у різних жанрах, включаючи аудіокниги, що робить її однією з найбільш універсальних платформ для обміну книг. Подібно до BookMooch користувачі отримують "кредити" за відправлення книг іншим членам і витрачають ці кредити на отримання нових книг. Серед недоліків, на які платформі варто звернути увагу є недостатня убезпеченість користувачів від

неправдивого опису книг. Оскільки платформа покладаєть саме на опис зареєстрованих читачів, стан книг може бути абсолютно різним і наявність самої книги також стоїть під питанням. Також дизайн сайту не відповідає вимогам користувачів по зрозумілості та зручності. Об'єднання цих проблем може призвести до краху, як це трапилось з платформою BookMooch [12]. Проте в цілому, на сьогоднішній день сервіс доволі популярний і веде свою діяльність в соціальних мережах, що вирізняє його серед конкурентів. Аналіз трафіку цього сайту показує, що він має 314,052 відвідування на місяць. Проте середній час відвідування сайту становить всього 3 хвилини 24 секунди. А 57.86% відвідувачів залишають сайт після перегляду лише однієї сторінки. Це говорить про те, що веб-сайт має помірне залучення користувачів з високим показником відмов, що може вказувати на проблеми з вмістом сайту або його навігацією.

Проаналізовано також платформу BookCrossing, котра трохи відрізняється функціоналом від вище згаданих. BookCrossing почав свою діяльність у 2001 році і з того часу став популярним у багатьох країнах, зокрема і в Україні, на відмінну від більшості програм з обміну книгами [12]. Унікальною концепцією цього сервісу є те, що книги "відпускаються" в громадських місцях для того, щоб інші могли їх знайти та прочитати, а потім знову "відпустити". Кожна книга має унікальний ID, який дозволяє стежити за її подорожами через спеціальний сайт, збільшуючи інтерес та залученість користувачів. Основною метою BookCrossing є популяризація читання і поділ книжок з широким колом людей. Вона також спрямована на створення глобальної спільноти любителів книг, які цінують ідею обміну знаннями та історіями через книги. BookCrossing побудував сильну міжнародну спільноту, яка ділиться книгами та досвідом читання. Проте, як і для інших платформ, є певні виклики. По перше, не завжди можливо контролювати, хто знайде книгу і коли це станеться і тому існує ризик, що книги можуть бути пошкоджені або втрачені. Також сервіс дуже залежить від кількості користувачів в регіоні. Оскільки в деяких регіонах чи спільнотах може бути менше учасників, що обмежує ефективність обміну. Проте сервіс взаємодіє з навчальними закладими та

іншими місцями де нові книги можуть бути цікавимо, що збільшує кількість учасників рециклінгу книг. Для розширення свого впливу BookCrossing може інтегрувати різноманітні сучасні технології. Загалом, BookCrossing є унікальною платформою, яка продовжує впливати на ринок обміну книгами, сприяючи культурі читання та ділення книгами на глобальному рівні [13].

У рамках критичного огляду програмних продуктів для обміну книгами, було розглянуто різні платформи, які пропонують користувачам унікальні способи взаємодії та обміну книгами. Ці платформи мають свої переваги та недоліки, що значно впливає на їхню популярність та залучення користувачів. Щоб краще зрозуміти слабкі та сильні сторони платформ було розроблену порівняльну таблицю(табл. 1.2) , в яку також внесено розроблений web - застосунок.

Таблиця 1.2

Порівняльна таблиця аналогів

Назва платформи	BookMooch	PaperBackSwap	BookCrossing	BookSwap
Основний функціонал	Обмін книгами за системою балів	Обмін книгами з використанням кредитної системи	"Випускання" книг у громадських місцях	Обмін книгами через пряме спілкування між користувачами
Рейтинг користувачів	+	+	+	+
Комунікація	Форум, приватні повідомлення	Форум	Форум, обмін повідомленнями електронною адресою	Чат між користувачами
Можливість продажу	-	-	-	+
Список бажаних книг	+	+	-	+

Загалом, успіх платформи для обміну книгами залежить від зручності її інтерфейсу, відповідності потребам користувачів, активності та залученості спільноти, а також ефективного управління ресурсами і технічної підтримки. Успішна платформа має забезпечувати не тільки можливість ефективного обміну, але й підтримувати високий рівень задоволення користувачів через прозорість, доступність та взаємну вигоду.

1.3 Вивчення ІТ-засобів, придатних для реалізації веб-застосунку

Технологічні інструменти, які використовуються для створення веб-застосунків, мають значний вплив на їх розвиток. Ця опція впливає на продуктивність розробки, масштабування проектів, а також на зручність подальшого обслуговування програмного забезпечення. C# - одна з важливих мов програмування, яка отримала широке визнання в галузі. Через переваги, які будуть розглянуті далі, було обрано саме цю мову програмування та відповідні технології для створення веб-застосунку.

Мова C# лідирує серед мов для платформи .NET - вільної, кросплатформенної, відкритої середовища розробки. Програми на C# можуть бути використані на широкому спектрі пристроїв, від IoT до хмарних сервісів, усюди, де це можливо. За допомогою цієї мови можна створювати програми для мобільних телефонів, персональних і портативних комп'ютерів, а також серверів.

C# - це універсальна кросплатформенна мова програмування, яка дозволяє розробникам ефективно писати високоефективний код. Завдяки мільйонам розробників, C# залишається найпопулярнішою мовою .NET. Вона має широку підтримку в екосистемі та всіх робочих навантаженнях .NET. Заснована на об'єктно-орієнтованих принципах, вона містить у собі багато можливостей з інших парадигм [14].

У цій роботі було вирішено використовувати потужний інструмент для створення RESTful веб-сервісів - ASP.NET Web API. Цей вибір був зумовлений необхідністю швидкої та гнучкої підтримки мережевих інтерфейсів для різних клієнтських застосунків, таких як мобільні застосунки та веб-браузери. ASP.NET Web API відзначається простотою у створенні веб-сервісів, які можуть приймати HTTP-запити з будь-яких пристроїв, спрощуючи інтеграцію та співпрацю з іншими системами та сервісами [15].

Даний інструмент дозволяє розробляти програми, які зручно масштабуються відповідно до збільшення вимог користувачів. Використання ASP.NET Web API дозволило оптимізувати комунікацію між сервером та клієнтом, зменшити навантаження на сервер і підвищити продуктивність застосунку. Також, ця технологія допомогла забезпечити безпечне з'єднання клієнтів і сервера, що є важливим для захисту конфіденційності та цілісності даних користувачів.

Іншим ключовим аспектом створення проекту був вибір систем управління базами даних, де б зберігалась вся неохідна інформація. Проаналізувавши такі бази даних як MySQL, SQL Server та PostgreSQL було обрано SQL Server. Цей вибір був зумовлений потребою в надійній, масштабованій та високопродуктивній СУБД, яка могла б забезпечити ефективну роботу з великими обсягами інформації [16].

Microsoft SQL Server відомий своєю високою продуктивністю, розширеними можливостями аналізу даних та глибокою інтеграцією з іншими продуктами Microsoft, що робить його ідеальним вибором для розробників, які працюють в середовищі .NET. Використання SQL Server дозволило оптимізувати доступ до даних та їх обробку, забезпечивши швидке виконання запитів і доставку даних в режимі реального часу для кінцевих користувачів мого застосунку.

Однією з основних переваг SQL Server є його мова запитів - Transact-SQL (T-SQL), яка є вдосконаленням стандарту SQL, спеціально призначеним для використання у SQL Server. T-SQL надає більше можливостей для програмування та керування даними, такі як процедурне програмування, тригери та автоматичне

керування транзакціями. Ці можливості допомагають розробникам керувати обробкою даних більш гнучко та ефективно, що важливо для стабільності та надійності веб-застосунків [17].

Для зручної взаємодії із системою управліннями базами даних було використано SQL Server Management Studio [18] (SSMS) — це інтегроване середовище для керування інфраструктурою SQL Server, яке забезпечує інструменти для налаштування, моніторингу та адміністрування екземплярів SQL Server. SSMS - невід'ємний інструмент для фахівців з баз даних, оскільки воно спрощує управління структурою бази даних, налаштуваннями безпеки та складними запитами.

Переходячи до розгляду фронтенд-частини веб-застосунку, було вирішено використовувати React - це бібліотека JavaScript, яка використовується для створення користувацьких інтерфейсів (UI) і визначає зовнішній вигляд веб-застосунку. React має відкритий вихідний код, створений компанією Facebook, яка активно співпрацює зі спільнотою для його розвитку.

React не лише підвищує естетичність сайту, але й покращує його продуктивність. Віртуальний DOM відповідає за це, і саме це є ключовою перевагою бібліотеки. За допомогою нього сайт може оновлювати лише необхідні частини сторінки. У web-застосунках зберігається структура попередньої версії для порівняння з новим станом інтерфейсу. Також важливий процес рендерінгу на серверній стороні. У реальності, як швидко завантажуються компоненти веб-застосунку, не буде залежати від потужності пристрою, на якому користувач відкриває його. Також, Реакт може бути використаний для різних нетрадиційних завдань, окрім конвертації HTML у відображення в браузері. За його допомогою можна створювати анімаційні графіки або зробити застосунок ізоморфним — для можливості серверу рендерити сторінки для початкового завантаження, щоб користувач бачив вміст, а не просто «Завантаження» [19].

1.4 Специфікація мети та завдань наукового дослідження

Базуючись на глибокому аналізі предметної галузі, критичному огляді існуючих програмних рішень та вивченні сучасних ІТ-засобів, що були детально розглянуті у попередніх підпунктах, були специфіковані цілі дослідження. Викладені вище розділи надали розуміння технічних можливостей і потреб користувачів, а також виявили прогалини в наявних рішеннях для обміну друкованими книгами. На основі цього аналізу було сформульовано мету дослідження та конкретизовано основні завдання, які необхідно вирішити для створення веб-застосунку, що відповідає сучасним вимогам індустрії та споживачів.

Мета і завдання, окреслені у цій частині, відображають прагнення вирішити ключові проблеми, виявлені під час попередніх етапів дослідження, та втілити в життя інноваційні ідеї для покращення взаємодії користувачів з книжковими ресурсами. Завдяки ретельно розробленому плану дій, наступні кроки дослідження дозволять не тільки створити ефективний веб-застосунок для обміну книгами, але й сприяти розвитку культури читання та відповідального споживання в умовах цифрової епохи.

Метою кваліфікаційної роботи є створення веб-застосунку для ефективного обміну друкованими книгами. Застосунок буде розроблено з використанням сучасних технологій, забезпечуючи високий рівень користувацького досвіду, з інтуїтивно зрозумілим інтерфейсом та швидким доступом до необхідних функцій. Основна мета полягає у створенні платформи, що сприятиме екологічному та економічному способу читання книг, забезпечуючи їх більш тривалий життєвий цикл через обмін між користувачами.

Серед завданнь варто зазначити:

- Визначення функціональних і нефункціональних вимог до веб-застосунку, включно з основними сценаріями його використання.

- Проектування структури бази даних, дизайну інтерфейсу користувача, та взаємодії між компонентами системи.
- Розробка веб-застосунку з використанням C# для серверної частини та React для клієнтської, інтеграція з API для обробки даних і взаємодії з користувачами.
- Впровадження систем безпеки для захисту особистої інформації користувачів.
- Проведення тестування функціональності та безпеки застосунку, з використанням автоматизованих та ручних методів тестування.
- Оцінка користувацького досвіду на основі зворотного зв'язку від користувачів та подальше вдосконалення інтерфейсу та функціональності.

Результатом проекту має стати повноцінний веб-застосунок, який не тільки задовольнить потреби користувачів у обміні книгами, але й стане зразком ефективного використання цифрових технологій для підтримки сталого використання ресурсів. Застосунок спрямований на формування спільноти книголюбів, які активно діляться книгами та враженнями від читання, створюючи нову культуру обміну книгами.

Розробка такого веб-застосунку стане також внеском у розширення академічних досліджень у галузі веб-технологій і книжкового обігу, демонструючи, як технології можуть сприяти екологічним та культурним змінам у суспільстві.

2 МЕТОДОЛОГІЯ ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ

2.1 Розробка архітектурного підходу до створення системи

Перед початком розробки застосунку було обрано архітектуру програми. Важливим аспектом у виборі було забезпечення гнучкості, масштабованості та легкості підтримки системи. Вирішивши орієнтуватися на модернізованість і адаптивність, було обрано трирівнева архітектуру, яка чудово підходить для веб-застосунків, таких як платформа для обміну друкованими книгами.

Трирівнева архітектура – це найпоширеніша архітектура взаємодії в інтернеті, її структура відображена на рис.2.1. Вона з'явилася, коли серверну частину дворівневої архітектури розділили на дві частини: шар логіки та шар даних. [20] Ця архітектура розподіляє застосунок на три рівні: рівень представлення (фронтенд), бізнес-логіку (серверний API) і базу даних. Ця розділення гарантує більшу структурованість коду і зручність в управлінні змінами, що є ключовими для довгострокової підтримки та розвитку системи [21].

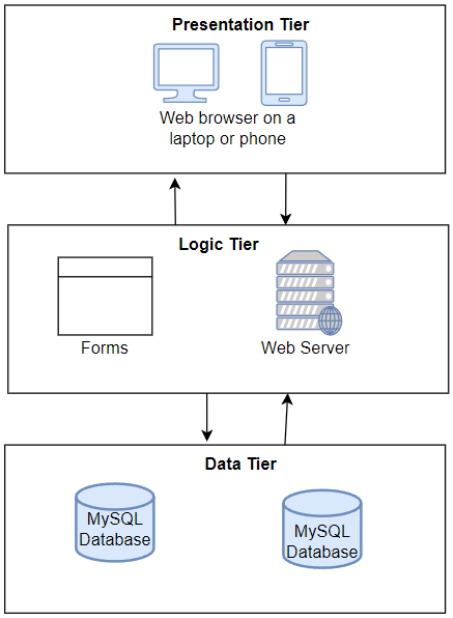


Рис. 2.1 Трирівнева архітектура

Рівень представлення відповідає за взаємодію з користувачем. Цей рівень не повинен містити бізнес-логіку та зберігати критично важливі дані. Також він не повинен взаємодіяти з шаром бази даних безпосередньо, а лише через бізнес-логіку. Однак якась логіка тут все ж є. По-перше, це взаємодія з користувачем через інтерфейс, валідація даних, що вводяться ним, робота з локальними файлами. Ще сюди можна віднести все, що стосується авторизації користувача та шифрування даних під час роботи з сервером. Ця частина проекту реалізована мовою React, що дозволяє створювати інтуїтивно зрозумілі та візуально привабливі користувацькі інтерфейси.

Наступний рівень - це бізнес-логіка. В цьому шарі відбувається вся обробка запитів/відповідей, а також логічні операції: математичні обрахунки та операції з даними. Він виконує роль посередника між рівнями відображення та доступу до даних, обробляючи та перевіряючи введену користувачем інформацію. Цей рівень впроваджено з використанням ASP.NET Web API, яке дозволяє ефективно обробляти запити від клієнтів і взаємодіяти з базою даних. Використання мови програмування C# на цьому рівні дозволяє використовувати потужні функції для створення складної бізнес-логіки. На рис. 2.2 зображена структура досліджуваного проєкту.

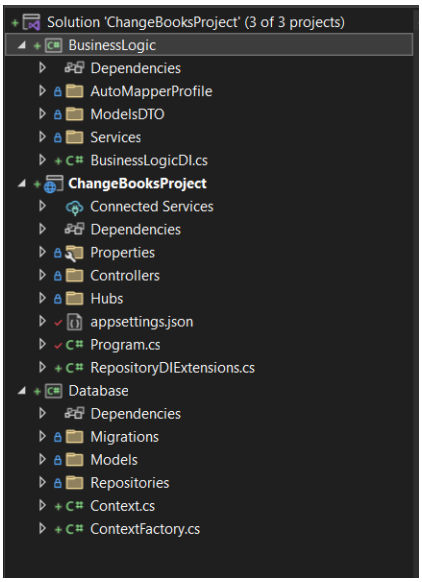


Рис. 2.2 Структура рівню бізнес логіки

Важливою задачею рівня бази даних є забезпечення збереження даних, які сервер зберігає для подальшого використання. Також забезпечується цілісність даних за допомогою зовнішніх зв’язків та ключів. На рівні бази даних також можна реалізовувати деяку бізнес-логіку, яка не потребує використання зовнішніх джерел даних окрім самої бази даних та її таблиць.

Рівень бази даних в проекті використовує SQL Server, щоб забезпечити надійне зберігання даних і швидкий доступ до них. База даних спроектована для підтримки великих обсягів роботи та надання швидких відповідей на запити користувачів. Всі її таблиці є на рис.2.3.

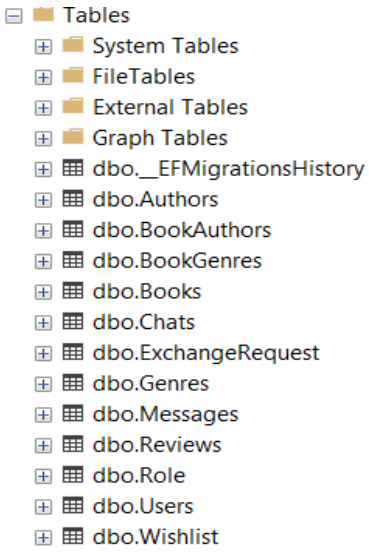


Рис. 2.2 Таблиці баз даних

2.2 Моделювання функціональних та нефункціональних вимог до системи

Наступним кроком при розробці веб-сайту по обміну друкованими книгами було визначення функціональних та нефункціональних вимог до системи, але перш ніж почати розглянемо що це таке.

Функціональні вимоги - це опис того, як система повинна поводитися. Вона визначає, які дії потрібно виконати системі для задоволення потреб або очікувань

користувача. На основі цього веб-застосунку було визначено такі функціональні вимоги:

- Аутентифікація – можливість створення нового облікового запису в базі даних.
- Авторизація – доступ до власного облікового запису, а також до різних функцій сайту.
- Перегляд книг – можливість пошуку та перегляду необхідних книг та отримання детальної інформації про неї.
- Створення книги – можливість створення книги у своєму обліковому записі у власній книгарні.
- Перегляд авторів – можливість пошуку та перегляду необхідного автора та отримання детальної інформації про нього.
- Список вподобаних книг – функція для створення власного списку бажаних книг, які користувач хоче здобути в майбутньому.
- Чат – можливість спілкування користувачів про книги або для уточнення щодо обміну/продажу книг.
- Профіль користувача – сторінка з особистими даними користувача з можливістю видалення та редагування її.
- Власна бібліотека з книгами – створення місця де користувач зможе додавати книги які він хоче обміняти/продати.
- Рейтинг та відгуки користувачів - система для оцінювання та відгуків про інших користувачів на основі їх активності на платформі.

Нефункціональні вимоги - це обмеження, накладені на систему, які визначають її атрибути якості. Нефункціональні вимоги важливі, оскільки вони допомагають забезпечити відповідність системи потребам користувача. На основі мого веб-застосунку було визначено такі функціональні вимоги:

- Безпека - забезпечення захисту інформації користувачів від несанкціонованого доступу шляхом шифрування переданих та збережених даних.
- Продуктивність – забезпечення швидкого реагування системи на запити користувачів.
- Гнучкість - архітектура системи повинна бути такою, щоб легко впроваджувати нові функції та покращення без значних перебоїв у роботі.

2.3 Створення проекту бази даних на основі аналізу інформаційних потреб

Першою дією при проектуванні бази даних було встановлення основних сутностей та їхніх взаємозв'язків, які відображають роботу веб-застосунку. Я виділив наступні основні сутності, які потрібно включити до бази даних:

- Users - користувачі

- Books - книги

- Authors – автори книг

- Genres – жанри книг

- Review – відгуки про користувачів

- Message – повідомлення, якими користувачі обмінюються в чаті

- Role – ролі користувача

- Wishlist – список вподобаних користувачами книг

В таблицях 2.1 – 2.8 наведено структури кожної з цих таблиць.

Таблиця 2.1

Структура таблиці “Users”:

Id	Int	Первинний ключ
UserName	String	Ім’я
Location	String	Місце знаходження
Description	String	Опис
PhoneNumber	String	Номер телефору
Image	Byte[]	Фото
Rating	float	рейтинг
Email	String	Емейл
OnlineTime	DateTime	Останній візит
PasswordHash	Byte[]	Хеш пароля
PasswordSalt	Byte[]	Зашифрований пароль
NumberOfExchange	Int	Кількість обмінів
RoleId	Int	Id ролі

Таблиця 2.2

Структура таблиці “Books”:

Id	Int	Первинний ключ
Title	String	Назва
Description	String	Опис
AnnouncedPrice	Decimal	Запропонована ціна
PageCount	int	Кількість сторінок
ConditionOfTheBook	String	Стан книги
Image	Byte[]	Фото
Language	string	Мова
OwnerId	Int	Id власника
DateTime	DateTime	Дата створення книги на сайті

Таблиця 2.3

Структура таблиці “Authors”

Id	Int	Первинний ключ
Name	String	Ім'я
Description	String	Опис
BDay	DateTime	Дата народження
DayOfDeath	DateTime	Дата смерті
Image	Byte[]	Фото

Таблиця 2.4

Структура таблиці “Genres”:

Id	Int	Первинний ключ
Name	String	Ім'я

Таблиця 2.5

Структура таблиці “Review”

Id	Int	Первинний ключ
Text	String	Текст
CreatedDate	DateTime	Дата створення
Rating	Float	Рейтинг
SenderId	Int	Id користувача, який створював відгук
RecipientId	Int	Id користувача, якому відгук призначений

Таблиця 2.6

Структура таблиці “Message”:

Id	Int	Первинний ключ
Content	String	Текст
Timestamp	DateTime	Час відправки
IsRead	Bool	Статус
SenderId	Int	Id користувача, який створював відгук
ReceiverId	Int	Id користувача, якому відгук призначений
ChatId	Int	Id чату

Таблиця 2.7

Структура таблиці “Role”:

Id	Int	Первинний ключ
Name	String	Ім'я

Таблиця 2.8

Структура таблиці “Wishlist”:

Id	Int	Первинний ключ
UserId	Int	Id користувача
BookId	Int	Id книги

На основі вищезазначеного аналізу розроблено ER-діаграму, зображену на рис.2.4, яка візуалізує структуру бази даних та взаємозв'язки між сутностями. ER-діаграма допомагає систематизувати дані і забезпечує зручність у подальшому розширенні та модифікації системи.

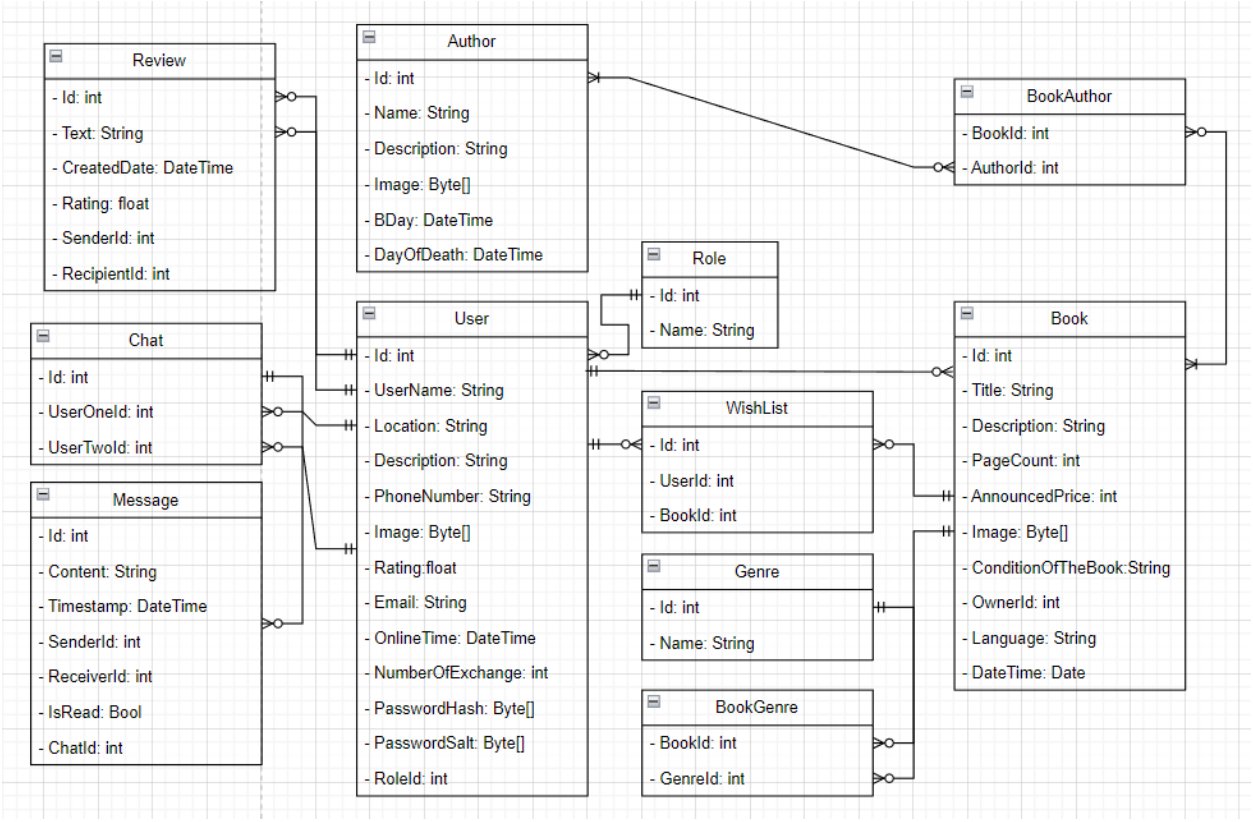


Рис. 2.4 Діаграма бази даних

Після детального аналізу структури бази даних, наступним кроком стало визначення ключових сценаріїв використання системи. Для наочного представлення цих сценаріїв було розроблено діаграму прецедентів, що ілюструє основні взаємодії користувачів з системою. Це дозволяє зосередитися на функціональних вимогах і визначити потенційні точки розширення системи. Діаграма прецедентів, представлена на рисунку 2.5, слугує ключовим інструментом для аналізу і проектування взаємодій користувачів з системою, сприяючи ефективній організації робочих процесів та оптимізації системних функцій.

Рис. 2.5 Діаграма прецедентів

2.4 Розробка концепції інтерфейсу користувача та сценаріїв взаємодії

Після створення структури бази даних наступним кроком у розробці веб-застосунку була розробка концепції інтерфейсу користувача. Цей процес включав визначення основних компонентів інтерфейсу, взаємодії між ними та розробку користувацьких сценаріїв, що відображають щоденне використання застосунку.

Для реалізації інтерфейсу використовувався інструмент Figma - онлайн-редактор, призначений для створення макетів, прототипів і дизайну користувальницького інтерфейсу (UI) і користувальницького досвіду (UX). Це потужний інструмент, який дає змогу дизайнерам працювати в режимі реального часу, спільно редагувати та ділитися проєктами в хмарі.

Використовуючи Figma було зроблено вайфрейми головних сторінок веб-сайту, з якими можна ознайомитись на рис.2.6-2.8 За допомогою них було визначено розташування головних компонентів інтерфейсу.

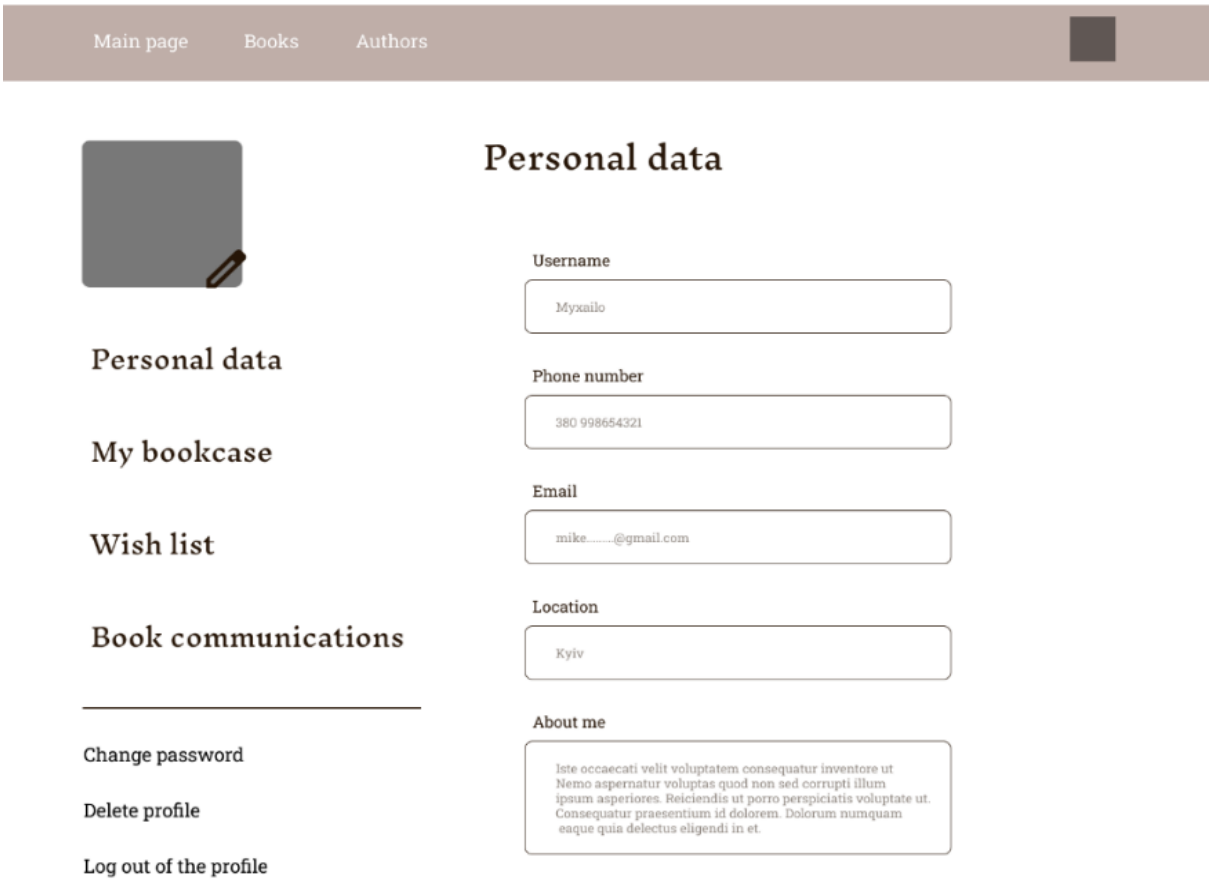


Рис.2.6 Вайфрейм профілю користувача

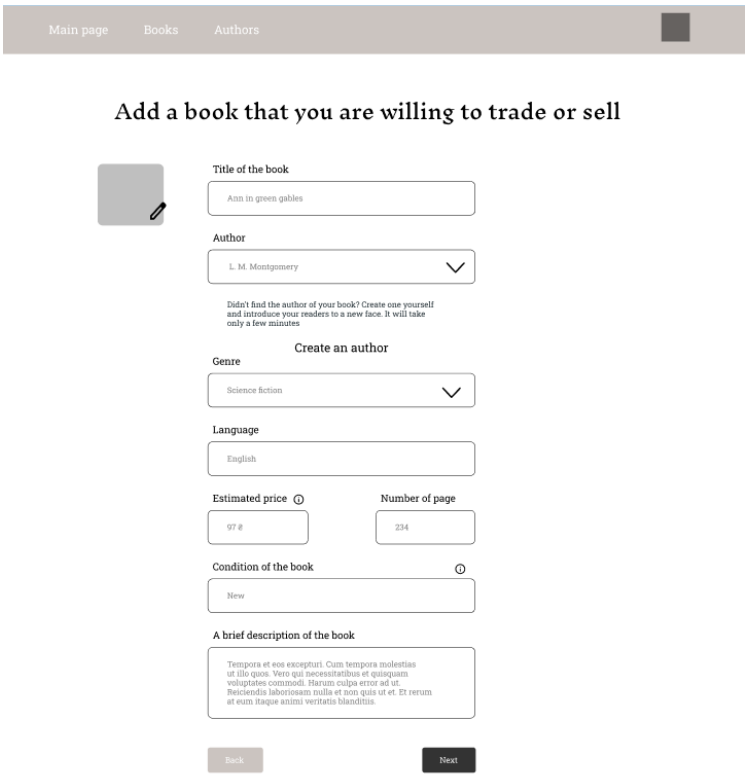


Рис. 2.7 Вайфрейм сторінки для додавання нової книги

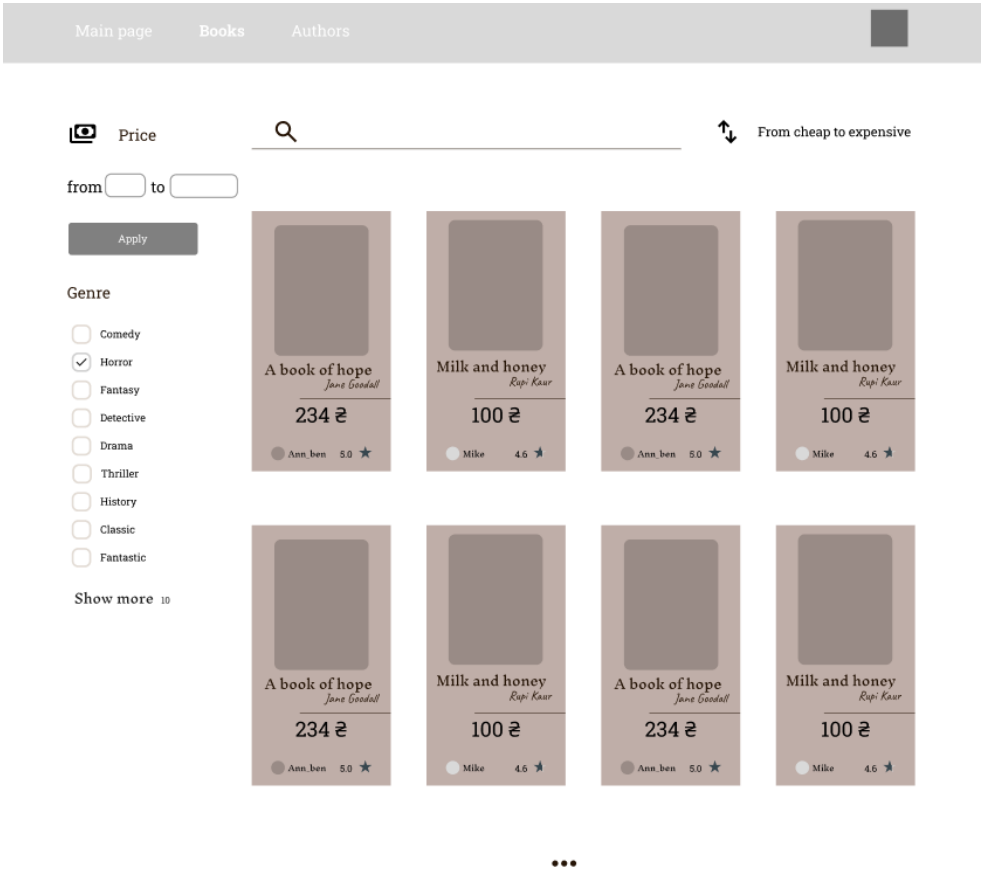


Рис. 2.8 Вайфрейм каталогу книг

Після цього було реалізовано більш деталізований дизайн інтерфейсу з визначенням колірної палітри, шрифтів та інших візуальних елементів. Зокрема, було ретельно продумано розміщення кнопок та інших інтерактивних елементів, щоб забезпечити логічність навігації та інтуїтивне розуміння функціоналу веб-сайту. На рис.2.9 – 2.11 представлені зображення з Figma, які ілюструють кінцеві варіанти інтерфейсу користувача.

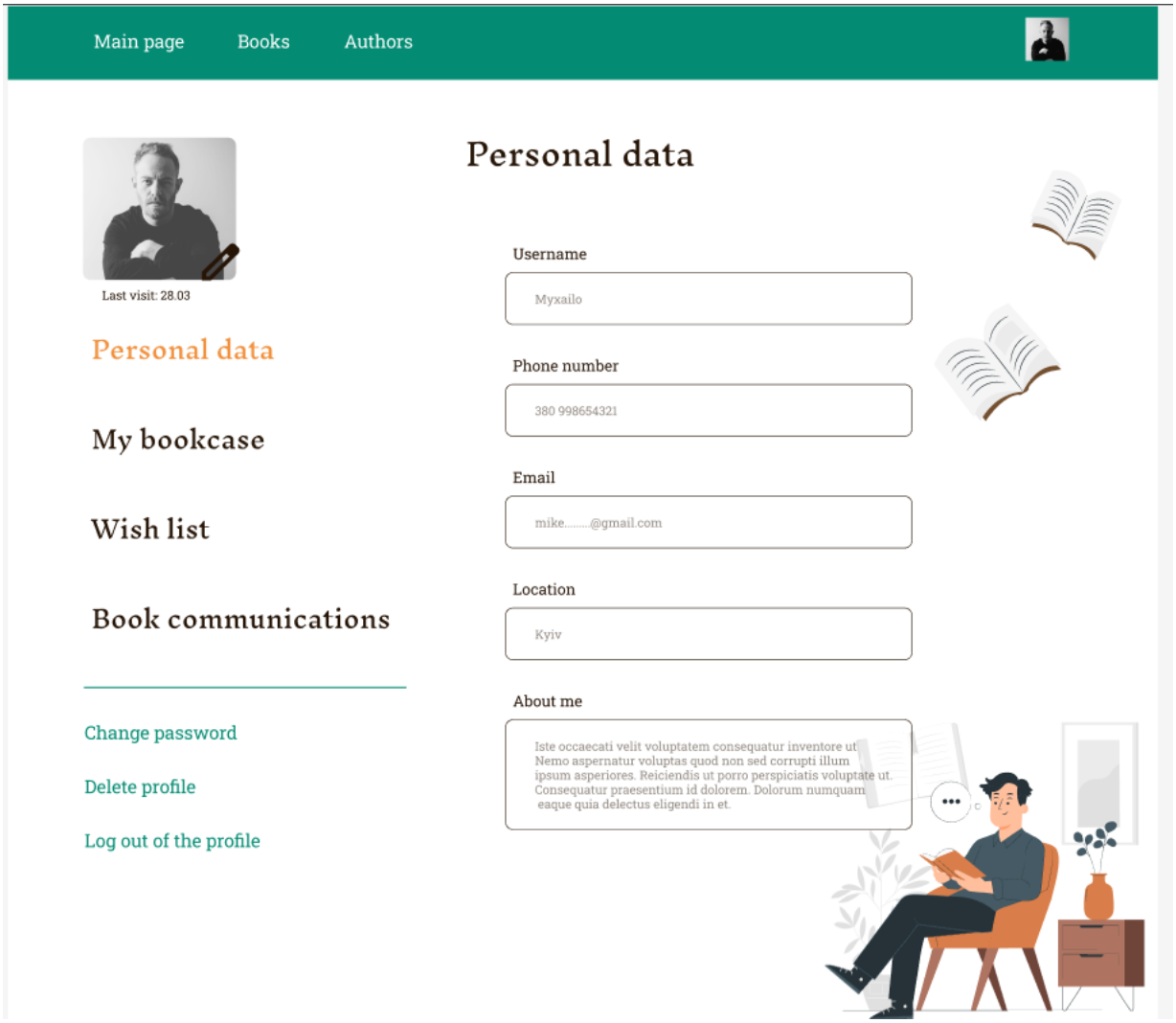


Рис. 2.9 Профіль користувача

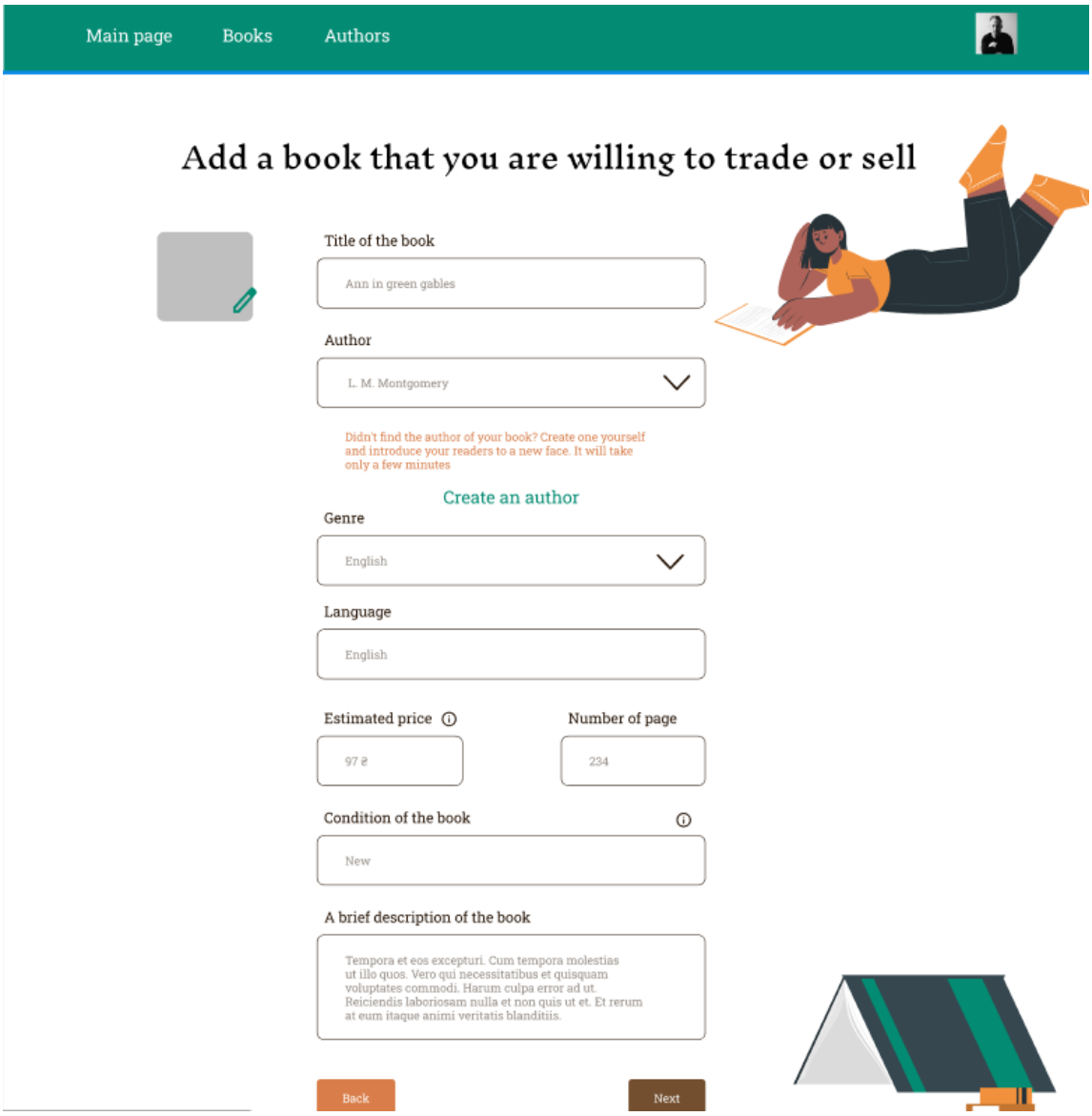


Рис. 2.10 Сторінка додавання книг

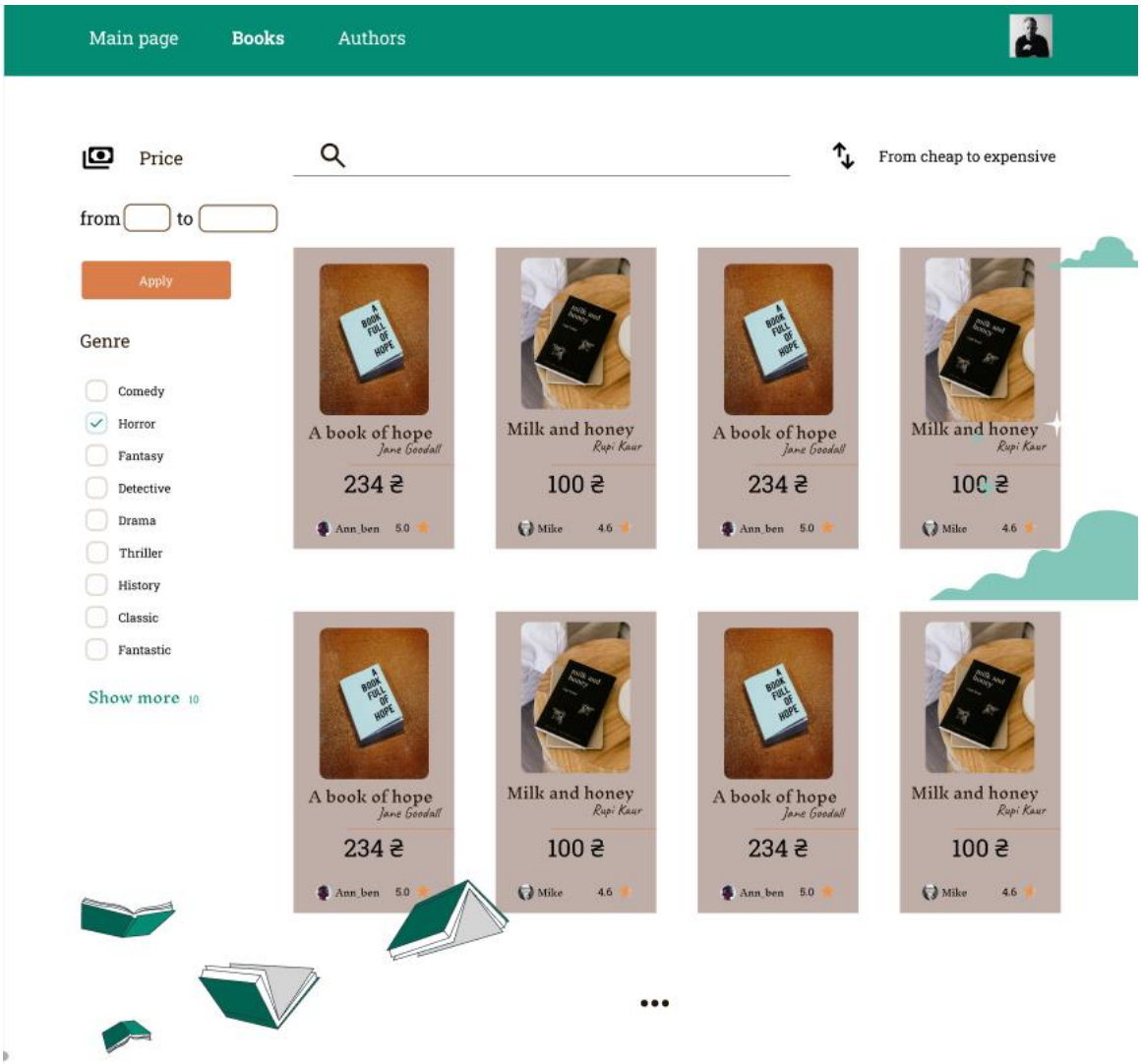


Рис. 2.11 Каталог книг

Після розробки архітектури та інтерфейсу користувача, було важливо систематизувати шляхи навігації та доступні функції веб-застосунку. Для цього було створено мапу сайту, яка відображає усі основні компоненти та їх взаємозв'язки в рамках платформи. Мапа сайту ілюструє структуру навігації та допомагає ідентифікувати ключові входи та виходи між різними частинами веб-сайту, що є незамінним інструментом для забезпечення логічної та інтуїтивно зрозумілої взаємодії користувачів з застосунком. Мапа сайту, представлена на рисунку 2.6, дає можливість візуально оцінити всі доступні функціональні можливості, їх розташування та шляхи доступу, сприяючи зручності навігації та оптимізації користувацького досвіду.

Рис 2.12 Мапа сайту

Процес розробки інтерфейсу користувача для веб-застосунку обміну книгами відіграє ключову роль у забезпеченні успішної взаємодії користувачів з платформою. Використання інструменту Figma для створення вайфреймів і прототипів дозволило ефективно розробити та візуалізувати розташування основних компонентів інтерфейсу, що сприяло ретельному плануванню користувацького досвіду. В результаті, були створені чіткі, інтуїтивно зрозумілі та естетично привабливі вайфрейми та кінцеві дизайнерські рішення, які забезпечують легку навігацію та високу ефективність користувацьких взаємодій.

Завдяки цій роботі було забезпечено логічну послідовність взаємодій і зрозумілість функціоналу застосунку, що є основою для забезпечення позитивного користувацького досвіду. Ці характеристики інтерфейсу важливі для залучення та утримання користувачів, а також для сприяння їх активному використанню веб-застосунку.

3 РЕАЛІЗАЦІЯ ТА ДЕМОНСТРАЦІЯ ФУНКЦІОНАЛЬНОСТІ ВЕБ- ЗАСТОСУНКУ

3.1 Розробка класів та методів: Структурне програмування системи

Серверна частина проекту реалізована за допомогою чотиришарової архітектури, що дозволяє чітко відокремити бізнес-логіку проекту, роботу користувацького інтерфейсу та бази даних, що дозволяє легке управління системи та впровадження нових функцій без редагування вже існуючих функцій.

Перший шар – це шар контролера, структура якого відображена на рис.3.1, який відповідає за взаємодію з користувачем та валідацію даних, що передає користувач. Цей шар приймає запити від користувача та передає керування наступному шару, і потім повертає результати користувачу.

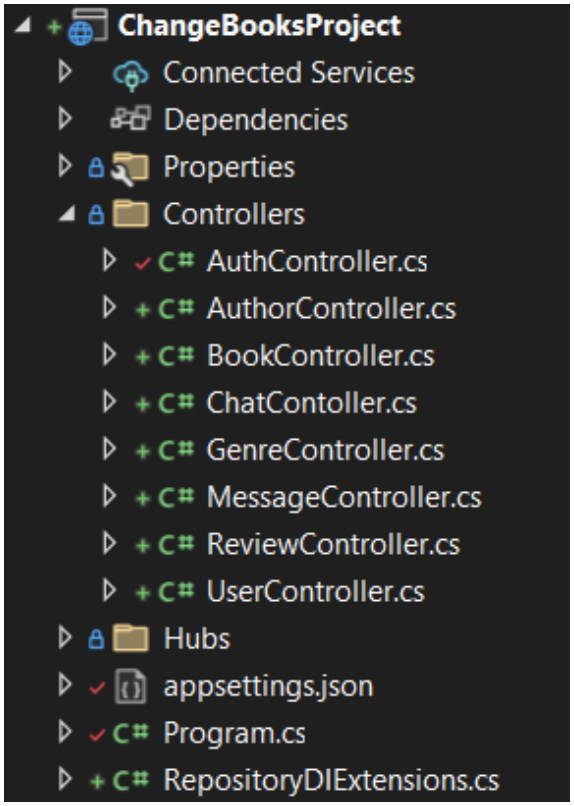


Рис. 3.1 Структура шару представлення

Для цієї роботи було створено такі контролери:

- AuthController – відповідає за керування авторизацією, аутентифікацією та створення користувачів.
- BookController – відповідає за керування операціями над книгами(створення, перегляд всіх книг, детальна інформація про одну книгу, видалення, редагування,
- AuthorController - відповідає за керування операціями над авторами книг(отримання даних про всіх авторів, детальна інформація про одного автора та створення).
- UserController – відповідає за керування операціями над користувачами(оновлення та видалення).
- MessageController - відповідає за повідомлення та обмін ними.
- ChatController – відповідає за керування чатами.
- ReviewController – відповідає за створення, редагування та видалення відгуками про користувачів.
- GenreController – відповідає за керування жанрами(отримання всіх жанрів).
- WishlistController – відповідає за керування списком бажань(отримання, редагування, видалення та створення)

Наступний шар – сервісний. Він є проміжним між шаром контролера та шаром доступу до даних. Основна роль сервісного шару полягає в упровадженні бізнес-логіки програми, що включає в управління бізнес-процесами, виконання складних операцій, які можуть вимагати додаткової логіки верифікації чи валідації та забезпечення взаємодії з шаром бази даних.

Інтерфейси сервісів є важливим елементом у цій архітектурі, бо вони встановлюють умови для сервісів, щоб забезпечити виконання конкретних обов'язків, які можна легко ідентифікувати та керувати. Кожен інтерфейс у проекті відповідає за виконання конкретної групи бізнес-логіки без прямого доступу до даних, які надають репозиторії. Інтерфейси служб, зображені на рис.3.2, такі як

IBookService, IUserService, IGenreService, IAuthorService, IReviewService, IMessageService, IWishlistService, IAuthService, IChatService визначають методи, які повинні бути реалізовані.

- BookService (залежний від IBookService) - відповідає за всі операції, пов'язані з книгами. Він обробляє додавання нових книг, оновлення існуючих записів, видалення книг та забезпечення доступу до детальної інформації про книги. Окрім основних CRUD операцій, BookService також виконує валідацію даних, перш ніж вони будуть внесені в базу даних, забезпечуючи, що всі дані відповідають бізнес-правилам та вимогам.
- UserService (залежний від IUserService) - відповідає за операції, пов'язані з користувачами, оновлення профілів користувачів, видалення користувацьких акаунтів і управління авторизацією. Він відповідає за забезпечення безпеки користувацьких акаунтів та їхніх даних.
- GenreService (залежний від IGenreService) - відповідає за управління жанрами книг. Цей сервіс дозволяє переглядати жанри.
- AuthorService (залежний від IAuthorService) - забезпечує управління інформацією про авторів. Це включає додавання нових авторів та доступ до списку всіх авторів.
- ReviewService (залежний від IReviewService) – відповідає за керування відгуками, які користувачі залишають для інших користувачів, а також обробляє створення, оновлення та видалення відгуків.
- MessageService (залежний від IMessageService) та ChatService (залежний від IChatService) - відповідають за обробку повідомлень та чатів у системі. Вони управляють створенням чат-сесій, відправленням та отриманням повідомлень.
- AuthService (залежний від IAuthService) – відповідає за керування аутентифікацією, авторизацією та забезпеченням безпеки сесій користувачів. Сервіс використовує різні технології для забезпечення безпеки, такі як токени доступу і шифрування паролів.

- WishlistService (залежний від IWishlistService) – відповідає за керування списком бажаних книг.

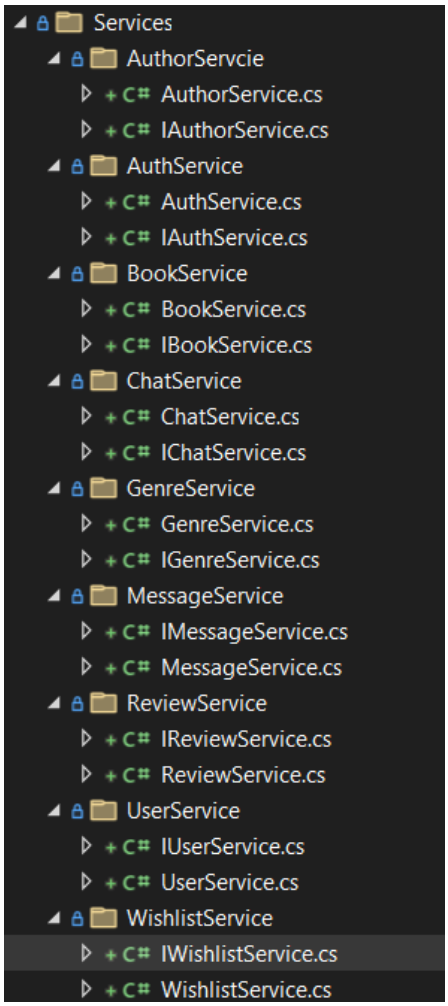


Рис. 3.2 Структура шару бізнес логіки

Шар репозиторіїв у проекті забезпечує абстракцію доступу до даних, ізолюючи бізнес-логіку від деталей реалізації збереження даних. Репозиторії служать як міст між доменною моделлю та базою даних, при цьому реалізуючи всі необхідні запити до бази даних.

При розробці цього шару проекту в першу чергу зосереджено на структурі та визначенні сутностей, які використовуються у базі даних. Ці сутності відображають основні об'єкти доменної моделі та їхні взаємозв'язки, які є критично важливими для бізнес-логіки мого додатку.

Були визначені такі сутності, зображені на рис.3.3:

- Book – книга включає такі поля як первинний ключ, назва, опис, кількість сторінок, зображення, ціна, стан, вторинний ключ власника, мова, час додавання книги на платформу.
- User – користувач включає такі поля як ім’я, місце знаходження, опис, номер телефону, рейтинг, кількість обмінів, останній візит, емейл, зображення, пароль.
- Author – автор книги включає такі поля як первинний ключ, ім’я, опис, зображення, дату народження, дату смерті.
- Genre – жанр включає такі поля як первинний ключ та ім’я.
- Review – відгук включає такі поля як первинний ключ, текст, дату створення та рейтинг.
- Chat – чат включає такі поля як первинний ключ та вторинні ключі користувачів у чаті.
- Message – повідомлення включає такі поля як первинний ключ, контент, вторинний ключ відправника, вторинний ключ отримувача, вторинний ключ чату та статус повідомлення.
- Wishlist – список бажань включає такі поля як первинний ключ, вторинний ключ користувача та вторинний ключ книги.
- UserReview – включає вторинний ключ відправника, ключ отримувача та слугує посередником між книгою та автором для зв’язку багато до багатьох.
- BookAuthor – включає вторинний ключ книги, вторинний ключ автора та слугує посередником між книгою та автором для зв’язку багато до багатьох.
- BookGenre – включає вторинний ключ книги, вторинний ключ жанру та слугує посередником між книгою та автором для зв’язку багато до багатьох.

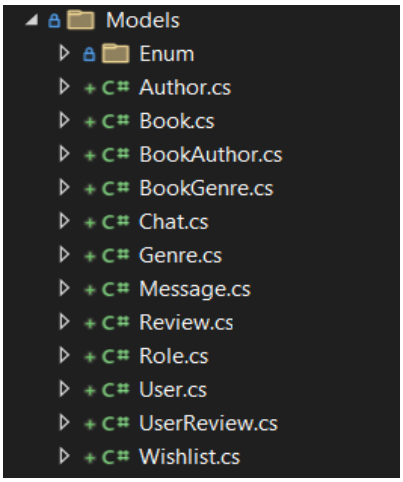


Рис. 3.3 Сутності використані в проєкті

Наступним кроком було створення репозиторіїв, які винесені на рис.3.4, кожен з яких відповідає за певну модель даних. Репозиторії, такі як BookRepository, UserRepository, GenreRepository, AuthorRepository, ReviewRepository, WishlistRepository, MessageRepository та ChatRepository, є інтегровані за допомогою визначених інтерфейсів, таких як IBookRepository, IUserRepository, IGenreRepository, IAuthorRepository, IReviewRepository, IWishlistRepository IMessageRepository, IChatRepository відповідно. Ці інтерфейси полегшують спілкування з базою даних і дозволяють зручно замінювати реалізацію репозиторіїв без впливу на інші компоненти системи.

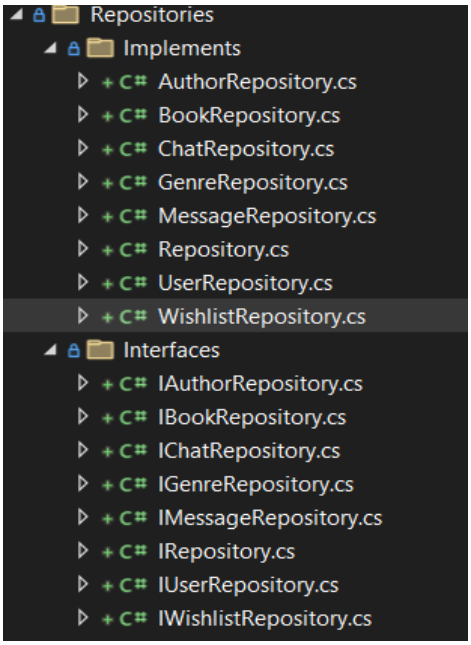


Рис. 3.4 Структура шару репозиторія

Останній шар проекту - шар бази даних. Цей шар відповідає за взаємодію з базою даних і забезпечує зв'язки між об'єктами в коді та структурами даних в базі. Він складається з контексту та фабрики контексту, які використовуються для керування взаємодією з базою даних через Entity Framework Core.

Клас Контекст (Context), зображений на рис.3.5-3.6, визначає основну сесію з базою даних. Він налаштовує структури та їх міжзв'язки, включаючи таблиці для користувачів, книг, авторів, жанрів, чатів, повідомлень, відгуків та списків бажань. Цей клас використовує DbSet для кожної сутності для керування операціями CRUD. Крім цього, він встановлює зв'язки між об'єктами і налаштування, такі як каскадне видалення, що дозволяє докладно контролювати поведінку бази даних під час взаємодії між об'єктами.

```
public class Context : DbContext
{
    1 reference
    public Context(DbContextOptions<Context> options) : base(options)
    {
    }
    6 references
    public DbSet<User> Users { get; set; }
    8 references
    public DbSet<Book> Books { get; set; }
    5 references
    public DbSet<Author> Authors { get; set; }
    0 references
    public DbSet<BookAuthor> BookAuthors { get; set; }
    0 references
    public DbSet<BookGenre> BookGenres { get; set; }
    2 references
    public DbSet<Genre> Genres { get; set; }
    0 references
    public DbSet<Review> Reviews { get; set; }
    3 references
    public DbSet<Message> Messages { get; set; }
    4 references
    public DbSet<Chat> Chats { get; set; }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Wishlist>()
            .HasOne(w => w.User)
            .WithMany(u => u.Wishlists)
            .HasForeignKey(w => w.UserId)
            .OnDelete(DeleteBehavior.Restrict);
        modelBuilder.Entity<Wishlist>()
            .HasOne(w => w.Book)
            .WithMany(b => b.Wishlists)
            .HasForeignKey(w => w.BookId)
            .OnDelete(DeleteBehavior.Restrict);
```

Рис. 3.5 Фрагмент коду налаштування відносин між сутностями в Entity Framework за допомогою Fluent API

```
        modelBuilder.Entity<UserReview>()
            .HasOne(ur => ur.Sender)
            .WithMany(u => u.SentReviews)
            .HasForeignKey(ur => ur.SenderId)
            .OnDelete(DeleteBehavior.NoAction);
        modelBuilder.Entity<UserReview>()
            .HasOne(ur => ur.Recipient)
            .WithMany(u => u.ReceivedReviews)
            .HasForeignKey(ur => ur.RecipientId);
        modelBuilder.Entity<BookGenre>()
            .HasKey(bg => new { bg.BookId, bg.GenreId });

        modelBuilder.Entity<BookGenre>()
            .HasOne(bg => bg.Book)
            .WithMany(b => b.Genres)
            .HasForeignKey(bg => bg.BookId);
        modelBuilder.Entity<BookGenre>()
            .HasOne(bg => bg.Genre)
            .WithMany(g => g.Books)
            .HasForeignKey(bg => bg.GenreId);
        modelBuilder.Entity<BookAuthor>()
            .HasKey(ba => new { ba.BookId, ba.AuthorId });
        modelBuilder.Entity<BookAuthor>()
            .HasOne(ba => ba.Book)
            .WithMany(b => b.Authors)
            .HasForeignKey(ba => ba.BookId);
        modelBuilder.Entity<BookAuthor>()
            .HasOne(ba => ba.Author)
            .WithMany(a => a.Books)
            .HasForeignKey(ba => ba.AuthorId);
        modelBuilder.Entity<Chat>()
           .HasKey(cr => cr.Id);

        modelBuilder.Entity<Message>()
            .HasKey(m => m.Id);
        modelBuilder.Entity<Message>()
            .HasOne(m => m.Chat)
            .WithMany(c => c.Messages)
            .HasForeignKey(m => m.ChatId)
            .OnDelete(DeleteBehavior.Cascade);
        modelBuilder.Entity<Chat>()
            .HasOne(c => c.UserOne)
            .WithMany()
            .HasForeignKey(c => c.UserOneId)
            .OnDelete(DeleteBehavior.Restrict);
        modelBuilder.Entity<Chat>()
            .HasOne(c => c.UserTwo)
            .WithMany()
            .HasForeignKey(c => c.UserTwoId)
            .OnDelete(DeleteBehavior.Restrict);
    }
}
```

Рис.3.6 Клас DbContext з визначеннями DbSet для сутностей і налаштуваннями моделі в методі OnModelCreating

ContextFactory, відображений на рис. 3.7, є класом, призначеним для створення екземпляру Context під час дизайну, зокрема для міграцій. Цей клас використовує підхід IDesignTimeDbContextFactory для створення контексту з певними параметрами, що спрощує налаштування та міграцію схеми бази даних.

```
namespace Database
{
    0 references
    public class ContextFactory : IDesignTimeDbContextFactory<Context>
    {
        0 references
        public Context CreateDbContext(string[] args)
        {
            var optionsBuilder = new DbContextOptionsBuilder<Context>();
            optionsBuilder.UseSqlServer("DB");

            return new Context(optionsBuilder.Options);
        }
    }
}
```

Рис. 3.7 Клас ContextFactory

3.2 Демонстрація роботи системи: аналіз взаємодій і інтерфейсу

У цьому підрозділі продемонстровано ключові аспекти взаємодії користувачів з веб-застосунком. Основні функції включають реєстрацію та авторизацію користувачів, перегляд, створення та управління книгами, створення та перегляд авторів, створення та редагування відгуків.

Реєстрація та авторизація, зображені на рис.3.8 – 3.10, є першими кроками взаємодії користувачів із системою. Ці функції гарантують безпечний доступ до всіх можливостей веб-застосунку. Під час реєстрації користувачі вводять інформацію для створення облікового запису і після цього входять у систему.

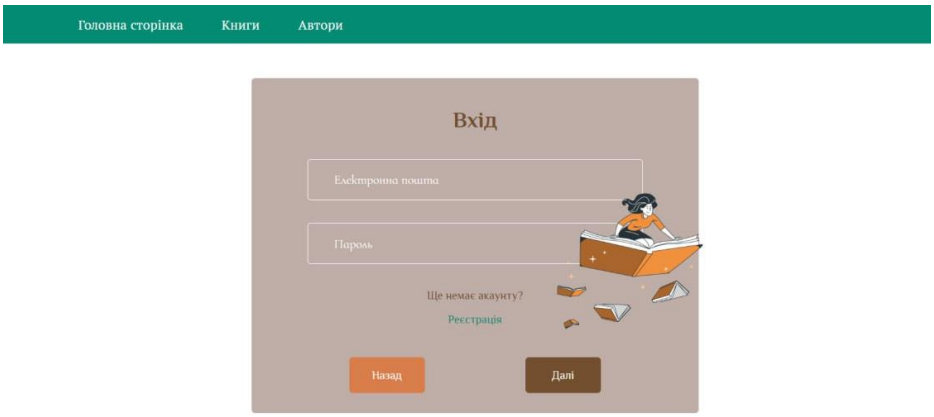


Рис. 3.8 Сторінка входу в профіль

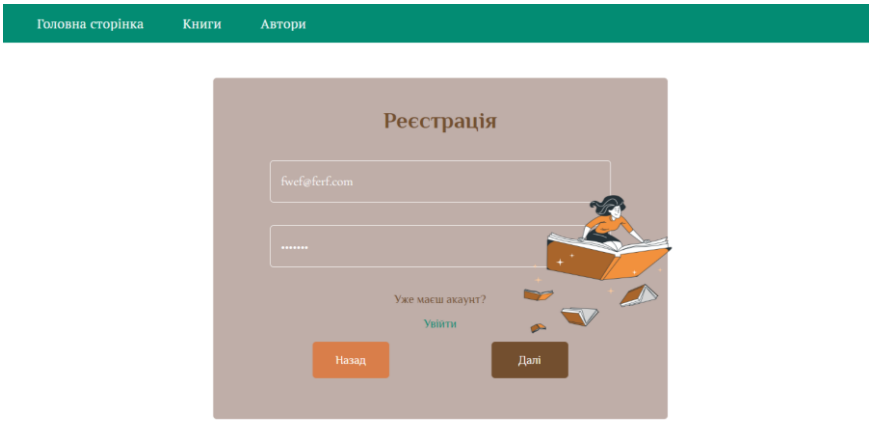


Рис. 3.9 Перша сторінка реєстрації

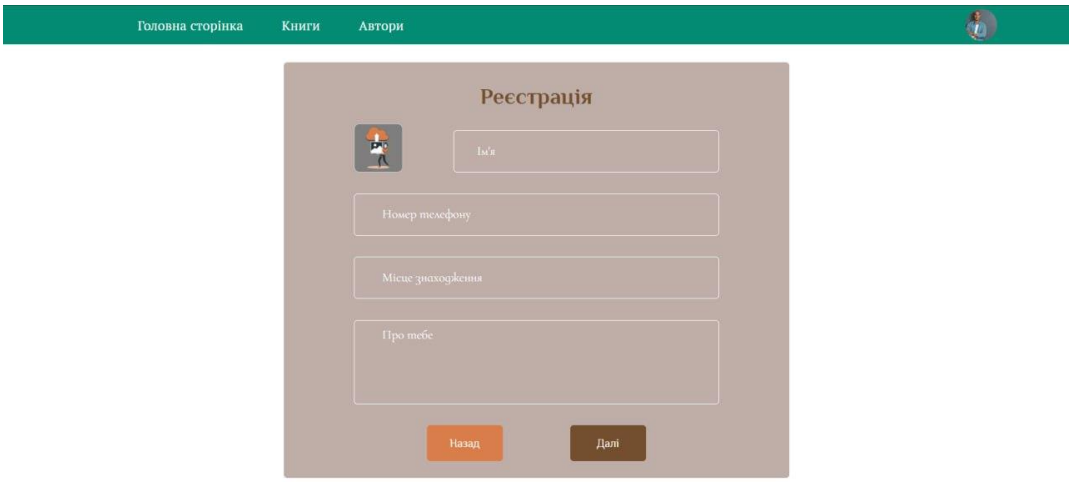


Рис. 3.10 Додаткова інформація при реєстрації

Після успішної авторизації користувач перенаправляється на сторінку свого профілю, де може переглядати та оновлювати свої особисті дані. Профіль користувача, на рис.3.11 надає зручний доступ до різних функцій взаємодії з системою, включаючи управління власними книгами та вибірками бажаних творів.

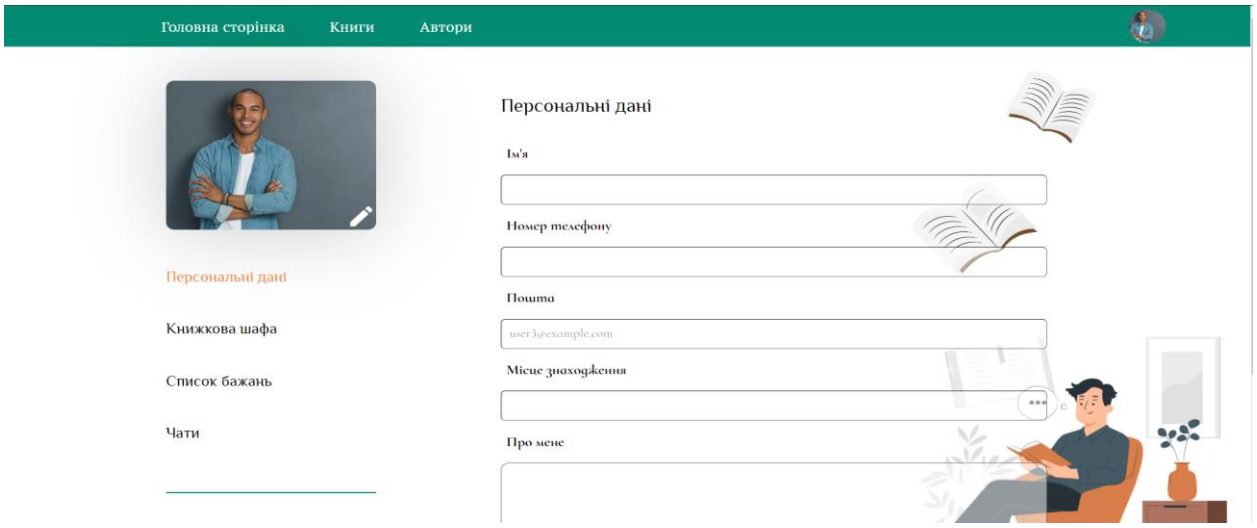


Рис. 3.11 Профіль користувача

В книжковій шафі користувач має змогу створювати книги, які він хоче обміняти або продати – рис.3.12- 3.13

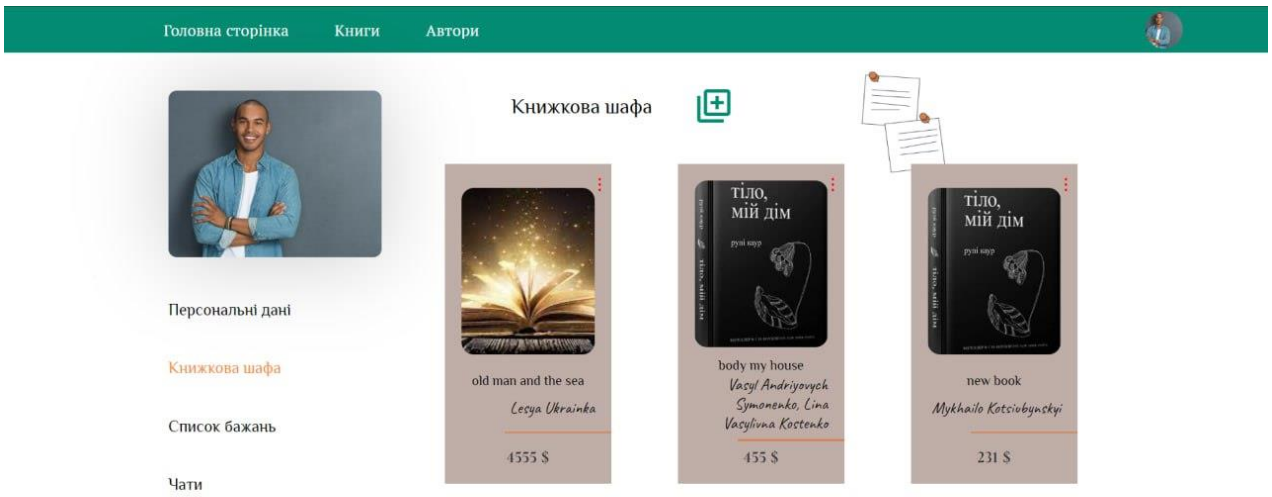


Рис. 3.12 Книги додані самим користувачем

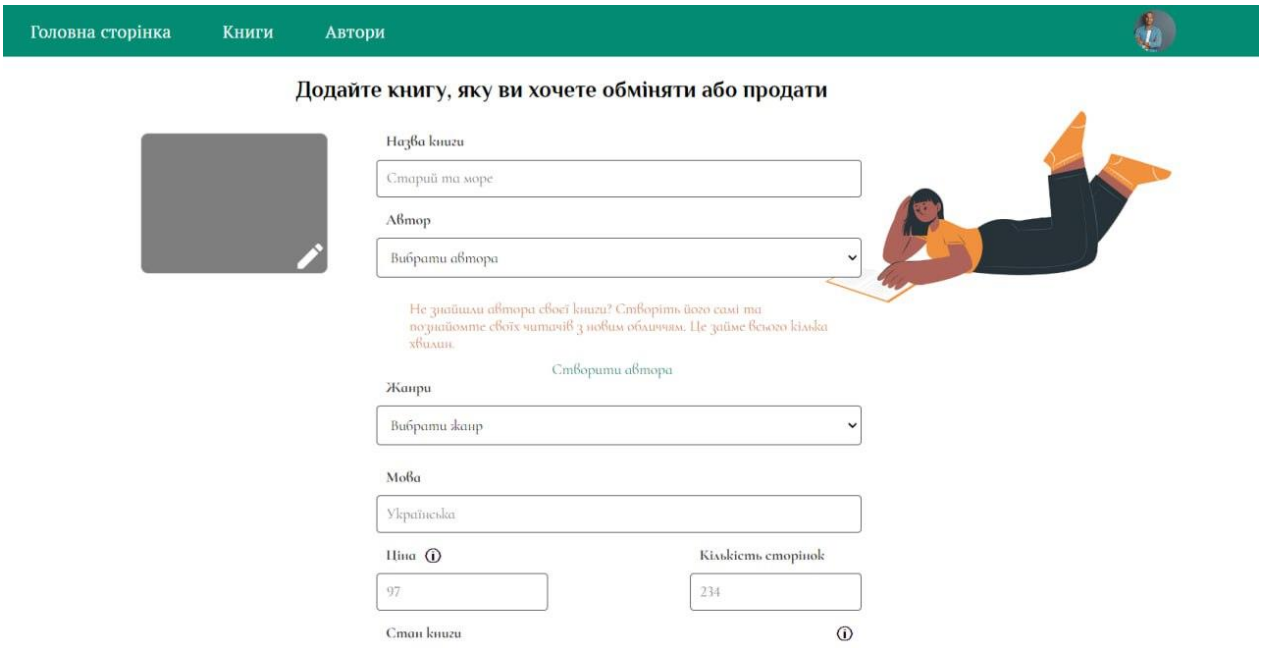


Рис. 3.13 Вікно додавання нової книги

А також можна додати необхідного автора, якщо його не було в списку запропонованих – рис.3.14

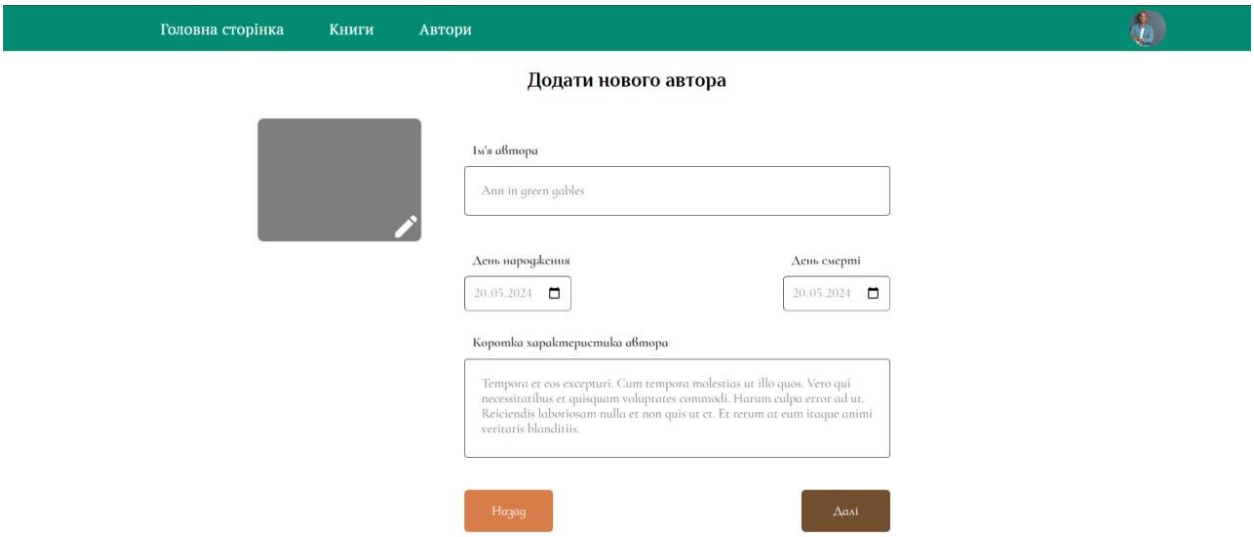


Рис. 3.14 Створення нового автора

Також зі сторінки профілю можна перейти на сторінки перегляду книг та авторів, візуал яких відображено на рис.3.15 -3.16

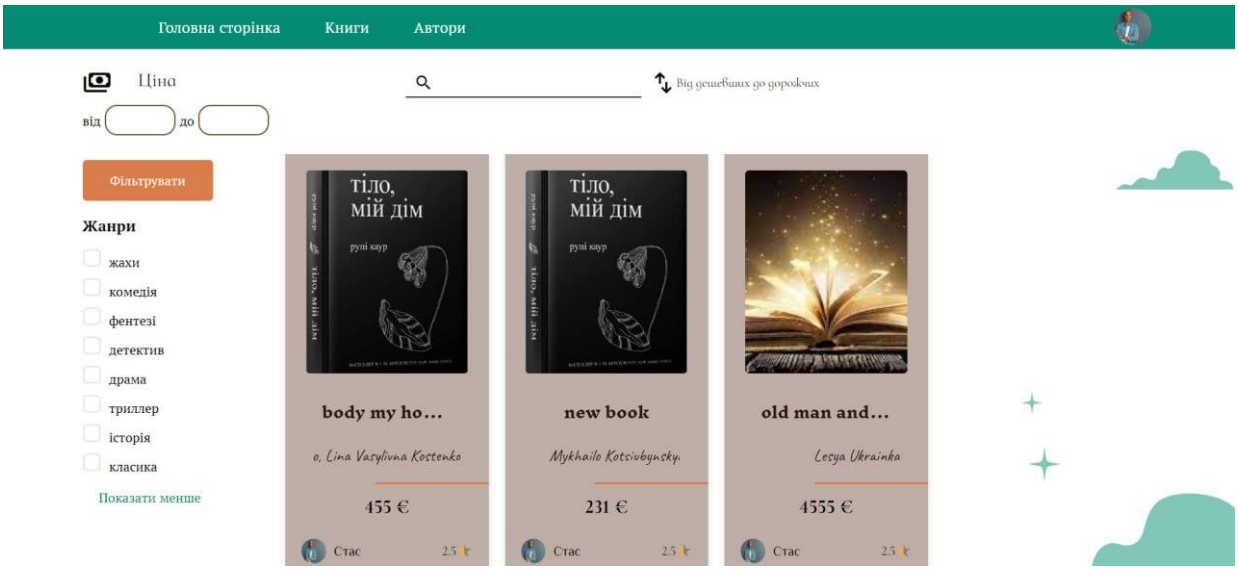


Рис. 3.15 Каталог книг

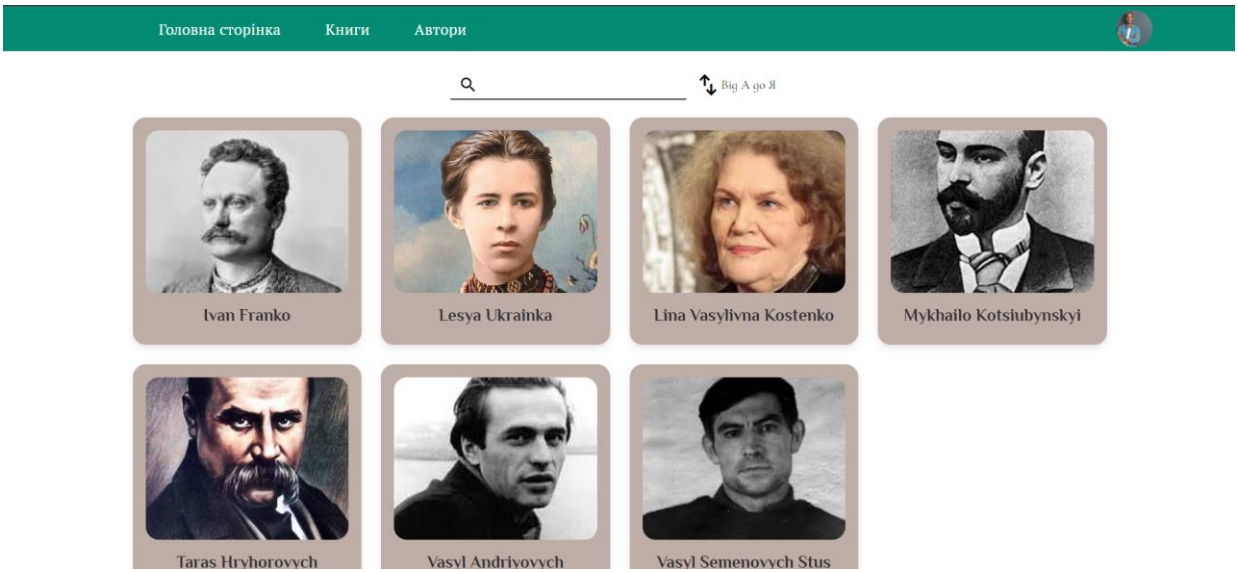


Рис. 3.16 Каталог авторів

Загальний аналіз взаємодії користувачів із веб-застосунком для обміну книгами виявив, що платформа ефективно виконує свої основні функції, забезпечуючи високий рівень користувацького досвіду. Реєстрація та авторизація користувачів здійснюються швидко та безпечно, що дозволяє легко інтегруватися в систему та одразу почати використання застосунку. Профіль користувача є центральним місцем для управління особистими даними та книжковими записами, що забезпечує зручний доступ до всіх необхідних функцій. Функціональність управління книгами дозволяє користувачам додавати, оновлювати або видаляти книги, що робить базу даних актуальною та корисною.

Перегляд каталогу книг та авторів виконаний таким чином, що користувачі можуть легко знаходити потрібну інформацію через зручно організовані інтерфейси.

Завдяки цим особливостям, веб-застосунок викликає позитивні враження у користувачів і підтримує ефективну взаємодію з системою. На основі зібраних даних можна зробити висновок, що платформа готова до широкого впровадження та активного використання.

4 ВЕРИФІКАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ВЕБ-ЗАСТОСУНКУ

4.1 Методологія та стратегії тестування програмного забезпечення

Розробка веб-застосунку для обміну книг вимагала уважної перевірки та оцінки різних складових системи. У зв'язку з масштабом проекту та важливістю покращення користувацького досвіду, було вирішено застосувати мануальне тестування як основний спосіб перевірки якості веб-застосунку. Цей підхід дозволяє ефективно відстежувати поведінку системи під час реальної взаємодії з користувачами, що має важливе значення для створення інтуїтивно зрозумілого та ефективного інтерфейсу.

Ручне тестування дозволило вивчити більшість аспектів веб-застосунку, включаючи процеси реєстрації та авторизації, управління профілем користувача, пошук і перегляд книг, спілкування з іншими користувачами через систему відгуків та чат. Виконано перевірку на помилки у валідації даних, функціональність навігаційних елементів інтерфейсу, а також загальну стабільність та продуктивність застосунку.

Мануальне тестування включало кілька ключових етапів:

1 Перевірка відповідності реалізованих функцій технічному завданню і вимогам користувачів.
2 Перевірка адаптивності дизайну під різні браузери.
3 Аналіз взаємодій між різними компонентами системи, зокрема, перевірка коректності взаємодії між клієнтом і сервером.
4 Забезпечення точності та надійності обробки даних у системі.

Було проведено перевірку функціональності пошуку та фільтрації книг, що відображено на рис.4.3. Це дозволило оцінити швидкість та точність обробки запитів користувачів системою і виявити можливі проблеми з продуктивністю.

Крім того, були випробувані можливості перегляду та керування профілями користувачів, додавання та видалення книг – рис. 4.1 - 4.2

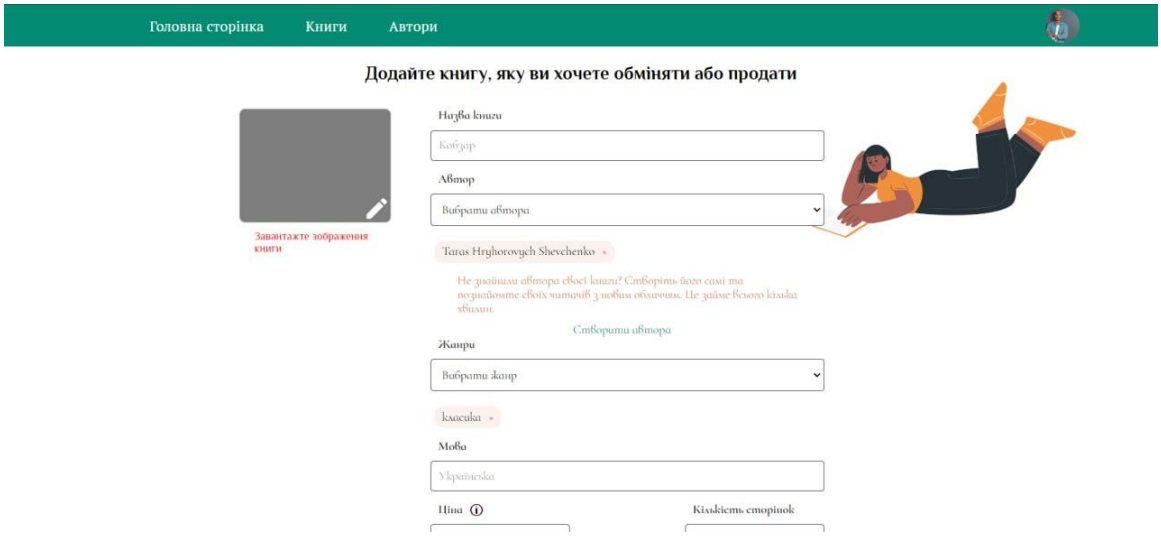


Рис. 4.1 Перевірка можливості створити книгу без додавання певних даних про неї (фото)

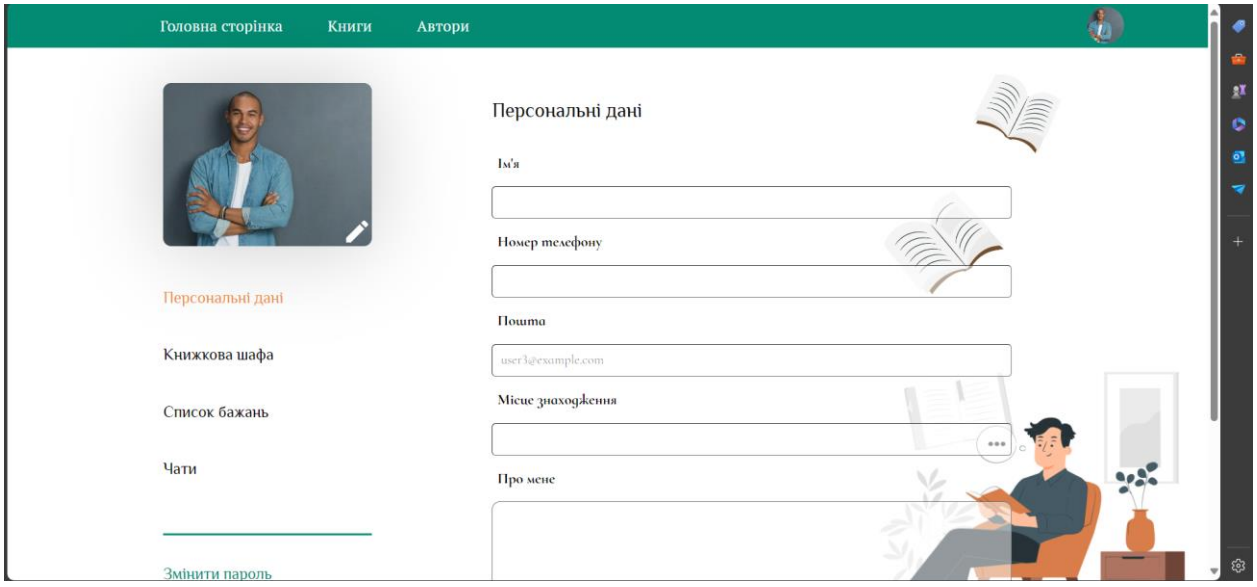


Рис. 4.2 Перевірка роботи web-застосунку в браузері Microsoft Edge

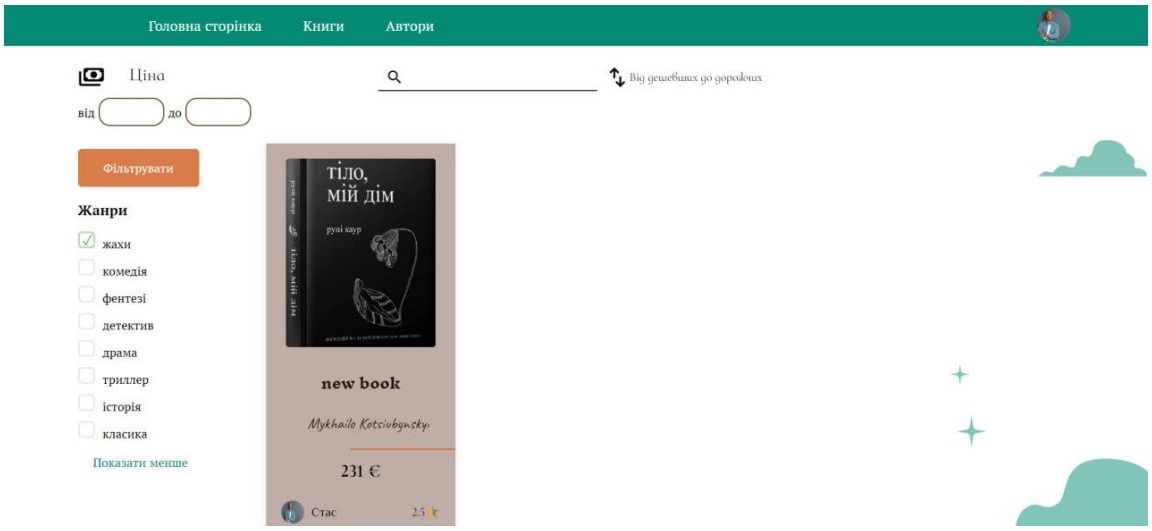


Рис. 4.3 Перевірка фільтрації книг

У рамках розробки веб-застосунку для обміну книгами було важливо забезпечити не тільки функціональність, але й високу якість користувацького інтерфейсу. Застосування мануального тестування дало можливість детально вивчити поведінку системи під час реальної взаємодії з користувачами. Цей підхід сприяв виявленню недоліків на ранніх стадіях розробки та забезпечив точне налаштування функцій відповідно до потреб користувачів.

Тестування кожної частини системи, від реєстрації користувачів до управління книжковими записами, дозволило ідентифікувати критичні точки збоїв і забезпечити стабільну роботу веб-застосунку. Перевірка адаптивності дизайну в різних браузерах гарантувала доступність і зручність застосунку для широкого кола користувачів. Через мануальне тестування я зміг не лише покращити інтерфейс та користувацький досвід, а й оптимізувати загальну продуктивність системи.

Важливою частиною процесу було також тестування інтеграції між різними компонентами системи, що допомогло усунути проблеми синхронізації та взаємодії між клієнтом і сервером. Аналіз функціональності пошуку та фільтрації книг показав, що система ефективно обробляє запити користувачів, забезпечуючи точні та швидкі результати.

Результати цього комплексного тестування дозволили зробити висновки про високу надійність розробленого веб-застосунку та його готовність до впровадження в експлуатацію. Таким чином, мануальне тестування виявилося незамінним інструментом у забезпеченні високої якості та ефективності програмного продукту.

4.2 Виконання тест-кейсів та аналіз результатів тестування

Після встановлення методології та стратегії тестування, було створено тест-кейси для перевірки функціональності веб-додатку. Кожен сценарій взаємодії користувача з системою мав свій відповідний тест-кейс і описував очікувані результати. Нижче перераховано основні тест-кейси, що були виконані під час проведення тестування:

Тест-кейс на реєстрацію та авторизацію користувачів, зображений на рис.4.4, включає ситуації введення як правильних, так і неправильних даних для перевірки системи валідації. Тест показав, що система правильно обробляє помилки, некоректне введення даних та забезпечує безпечний доступ до функцій веб-застосунку.

Рис. 4.4 Тест-кейс авторизаціцї користувачів

Тест-кейс управління книгами, відображений на рис.4.5 – 4.7 включає перевірку додавання нових книг, оновлення існуючих записів, видалення книг та отримання інформації про книги. Було перевірено, що всі операції виконуються коректно та відповідають бізнес-логіці.

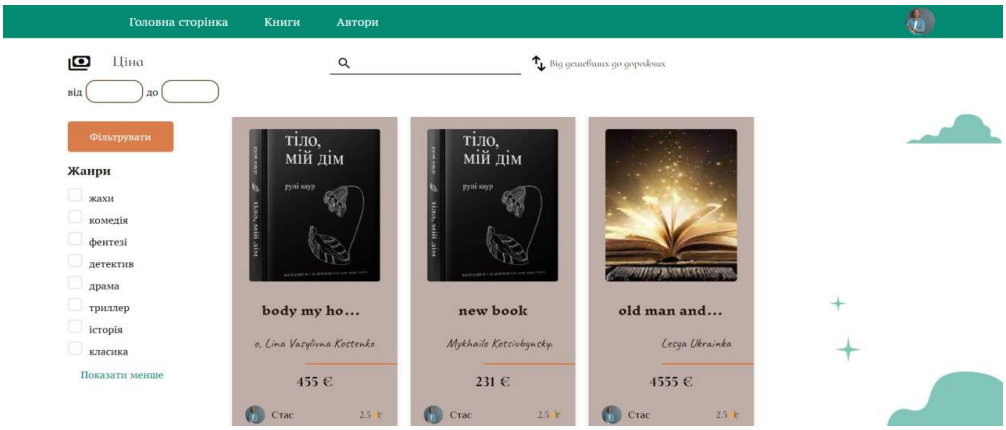


Рис. 4.5 Тест-кейс управління книгами (кількість наявних книг на платформі)

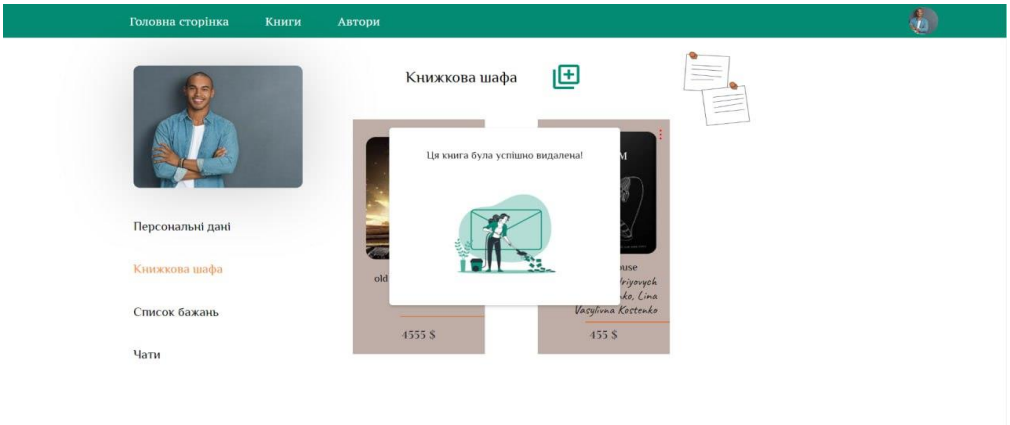


Рис. 4.6 Тест-кейс управління книгами (видалення книги користувачем)

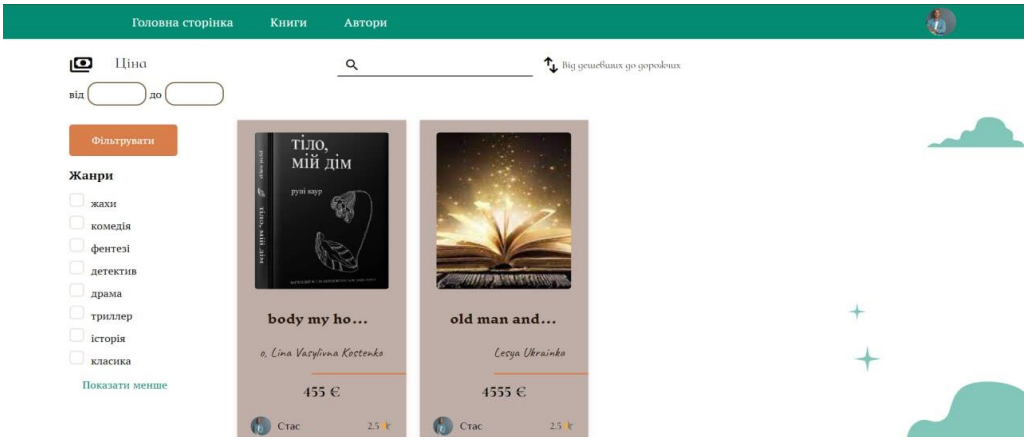


Рис. 4.7 Тест-кейс управління книгами (відображення меншої кількості книг, оскільки 1 вже була видалена)

Тест-кейс пошуку та фільтрації книг, відображений на рис.4.8, перевірявє параметри пошуку назви книг та жанри. Результати показали, що система швидко та точно обробляє запити, забезпечуючи користувачам зручний інструмент для пошуку необхідної інформації.

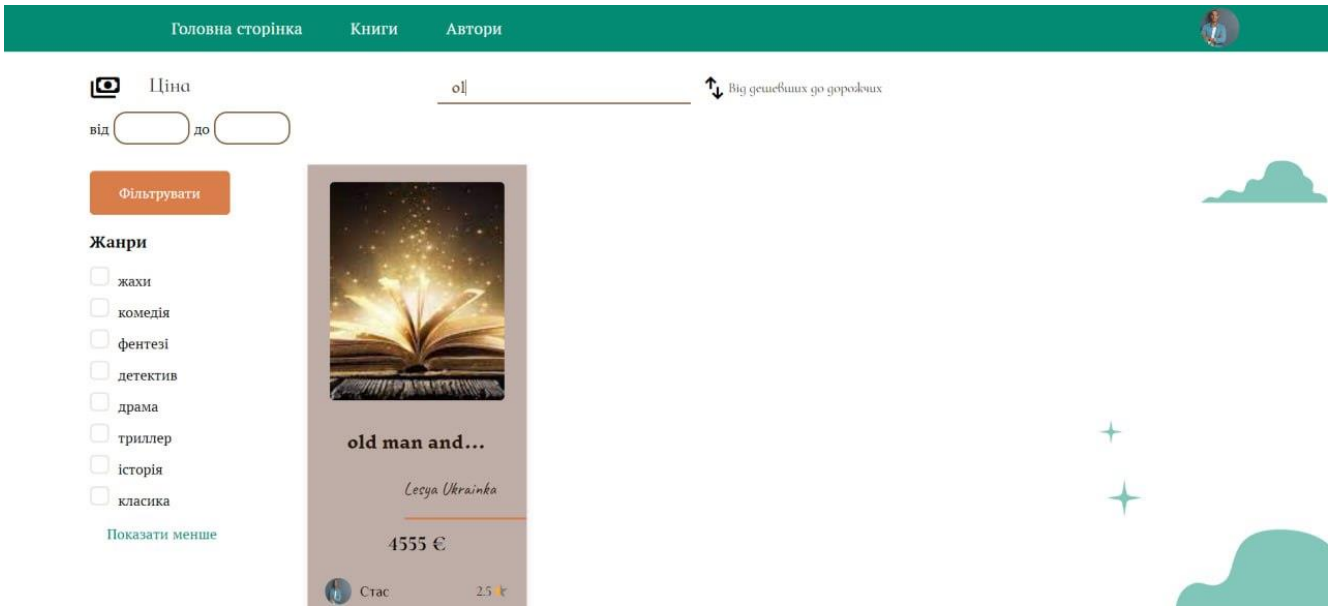


Рис. 4.8 Тест-кейс пошуку та фільтрації книг

Під час тестування основних функціональних можливостей веб-застосунку було проведено ряд тест-кейсів, які демонструють надійність та стійкість системи. Тести охопили ключові аспекти взаємодії користувачів із системою, включаючи процеси реєстрації та авторизації, управління книгами, а також пошук і фільтрацію книг. Результати тестування підтвердили, що система ефективно обробляє як валідні, так і невалідні вхідні дані, забезпечує коректність даних та відповідність бізнес-логіці в управлінні ресурсами, а також демонструє високу точність і швидкість у пошуку та фільтрації книг.

Ці результати вказують на високу якість розробленого програмного забезпечення і дають підстави для подальшого впровадження системи на реальних користувачах, що сприятиме покращенню їхнього досвіду користування та задоволенню їхніх потреб в доступі до інформаційних ресурсів.

ВИСНОВКИ

У результаті виконання дипломної роботи розроблено веб-застосунок для обміну друкованими книгами. Його актуальність полягає в тому, що цей ресурс значно полегшує процес пошуку та обміну книгами між користувачами.

1. Проведено огляд існуючих рішень та аналіз ринкових рішень показав, що багато існуючих платформ мають обмеження щодо функціональності, як от можливість продажу книги чи безпосередній обмін повідомленнями між користувачами.

2. Спроектовано архітектуру системи для веб-застосунку.

3. Розроблено схему бази даних системи обміну друкованими книгами.

4. Реалізовано інтерфейс користувача для веб-застосунку обміну книгами, який забезпечує зручне управління книгами та ефективне спілкування між користувачами.

5. Реалізовано основні функції веб-додатку, дотримуючись функціональних та нефункціональних вимог, таких як створення та керування особистим профілем, керування власними книгами та перегляд книг інших користувачів, перегляд авторів та створення необхідного автора, чат, список вподобаних книг, а також забезпечено захист від несанкціонованого доступу до даних користувача шляхом шифрування паролю користувача.

6. Застосунок проаналізовано за рахунок мануального тестування. В результаті тестування було виявлено значну затримку у відображенні книг, а саме 2.5 секунди, що вдалось вчасно виявити та знизити швидкість завантаження до 1 секунди.

7. Скворцов М.О., Замрій І.В. Оцінка ефективності баз даних для зберігання інформації про книги та користувачів у веб-додатку. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 112 - 114.

ПЕРЕЛІК ПОСИЛАНЬ

1. Замрій І.В., Скворцов М.О. Оцінка ефективності баз даних для зберігання інформації про книги та користувачів у веб-додатку «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2024. С.112-114.
2. Books Market Size, Share & Trend Analysis Report By Type (Science, Historical, Mystery, Fantasy, Literary, Contemporary/Realistic, Romance, Educational, Comic, Others), By Format, By Distribution Channel, By Region, And Segment Forecasts, 2024 - 2030. 2021. 83 p. URL: https://www.grandviewresearch.com/industry-analysis/books-market
3. Nuria Huete-Alcocer. A Literature Review of Word of Mouth and Electronic Word of Mouth: Implications for Consumer Behavior. 2017. URL: https://www.researchgate.net/publication/318684075_A_Literature_Review_of_Word_of_Mouth_and_Electronic_Word_of_Mouth_Implications_for_Consumer_Behavior
4. The Impact of Online Reviews on Consumers' Purchasing Decisions: Evidence From an Eye-Tracking Study. 2022. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9216200/
5. Data protection under GDPR and online privacy - Your Europe. URL: https://europa.eu/youreurope/business/dealing-with-customers/data-protection/index_en.htm
6. Веб-застосунок BookMooch. URL: http://bookmooch.com/
7. What caused BookMooch to fail?. ProAI | Pro Business Plans. URL: https://www.quora.com/What-caused-BookMooch-to-fail
8. Веб-застосунок GoodReads. URL: https://www.goodreads.com/
9. Number of registered members on Goodreads from May 2011 to July 2019 URL: https://www.statista.com/statistics/252986/number-of-registered-members-on-goodreadscom/

10. Madhav Ajayamohan. Do Goodreads really show “Good Reads”?. A Journal of First Year Writing at UTM. 2023. 11 May. URL: https://jps.library.utoronto.ca/index.php/writing/article/download/41706/31915
11. Google Analytics. URL: https://developers.google.com/analytics/learn
12. Веб-застосунок BookCrossing. URL: https://www.bookcrossing.com/
13. Explore the world of Bookcrossing. Distributed Computing Group Computer Engineering and Networks Laboratory ETH Zurich. 2011. P. 11–20. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3adfd3f4289890bcf4393140d832b69be97276a9
14. A tour of the C# language. Microsoft. 2024. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview.
15. Andrew Troelsen Philip Japikse. C# 6.0 and the .NET 4.6 Framework. 7th ed. 2015.
16. Andrew Troelsen, Philip Japikse. ASP.NET MVC and Web API. 2015.
17. Rafał Wodyk, Maria Skublewska-Paszkowska. Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework Porównanie wydajności relacyjnych baz danych SQL Server, MySQL oraz PostgreSQL z zastosowaniem aplikacji webowej i frameworku Laravel. Computer sciences institute. 2020. URL: https://ph.pollub.pl/index.php/jcsi/article/view/2279/2152
18. Itzik Ben-Gan. T-SQL Fundamentals URL: https://books.google.com.ua/books?hl=uk&lr=&id=W4zADAAAQBAJ&oi=fnd&pg=PT21&dq=t-sql&ots=t8WVcO8Brb&sig=zDMejA6-xAghgrlVdXUJ36EgEWw&redir_esc=y#v=onepage&q=t-sql&f=false
19. ІЩо таке SQL Server Management Studio (SSMS)? Microsoft. 23.05.2023 URL: https://learn.microsoft.com/ru-ru/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16
19. ІЩо таке Реакт?. Web Developer in UA. URL: https://web-developer.in.ua/assets/articles/react/react-introduction/react-introduction.html
21. N-Tier Architecture | Baeldung on Computer Science. Baeldung on Computer Science. URL: https://www.baeldung.com/cs/n-tier-architecture

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА WEB-ЗАСТОСУНКУ ПО ОБМІНУ ДРУКОВАНИМИ КНИГАМИ МОВОЮ C# ТА REACT

Виконав студент 4 курсу
групи ПД-41
Скворцов Михайло Олександрович
Керівник роботи
Д.т.н., доцент, завідувач кафедри ІПЗ Замрій Ірина Вікторівна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – спрощення процесу обміну друкованими книгами.

Об’єкт дослідження – обмін друкованими книгами.

Предмет дослідження – web-застосунок, спрямований на обмін друкованими книгами.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідження сучасних тенденцій та потреб у сфері обміну друкованими книгами.
2. Аналіз існуючих рішень на ринку, їхніх функціональних можливостей, переваг та недоліків.
3. Проектування архітектури веб-додатку, забезпечення його масштабованості, безпеки.
4. Проектування та розробка бази даних системи обміну друкованими книгами.
5. Проектування інтерфейсу користувача системи обміну друкованими книгами.
6. Розробка та імплементація основних функцій веб-застосунку обміну друкованими книгами.
7. Проведення мануального тестування системи на відповідність вимогам.
8. Проходження апробації на Науково-технічних конференціях.

3

АНАЛІЗ АНАЛОГІВ

Назва платформи	BookMooch	PaperBackSwap	BookCrossing	BookSwap
Основний функціонал	Обмін книгами за системою балів	Обмін книгами з використанням кредитної системи	"Випускання" книг у громадських місцях	Обмін книгами через пряме спілкування між користувачами
Рейтинг користувачів	+	+	+	+
Спосіб комунікації	Форум, приватні повідомлення	Форум	Форум, обмін повідомленнями електронною адресою	Чат між користувачами
Можливість продажу книги	-	-	-	+
Список бажаних книг	+	+	-	+

4

ВИМОГИ ДО ЗАСТОСУНКУ

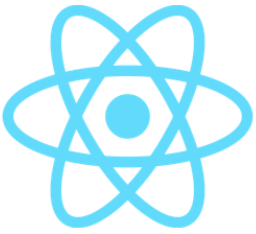
Функціональні вимоги:
1. Створення нового облікового запису в базі даних.
2. Доступ до власного облікового запису, а також до різних функцій сайту.
3. Пошук та перегляд необхідних книг та отримання детальної інформації про них.
4. Створення книги у своєму обліковому записі у власній книгарні.
5. Пошук та перегляд необхідного автора та отримання детальної інформації про нього.
6. Функція для створення власного списку бажаних книг, які користувач хоче здобути в майбутньому
7. Чат – можливість спілкування користувачів про книги або для уточнення щодо обміну/продажу книг.
8. Сторінка з особистими даними користувача з можливістю видалення та редагування її.
9. Створення місця де користувач зможе додавати книги які він хоче обміняти/продати.
10. Система для оцінювання та відгуків про інших користувачів на основі їх активності на платформі.

Нефункціональні вимоги:
1. Забезпечення захисту інформації користувачів від несанкціонованого доступу шляхом шифрування переданих та збережених даних.
2. Забезпечення швидкого реагування системи на запити користувачів – 1.5 секунди.
3. Архітектура системи повинна бути такою, щоб впроваджувати нові функції без значних змін в інших частинах застосунку та покращення без значних перебоїв у роботі.
4. Кросбраузерність.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Figma



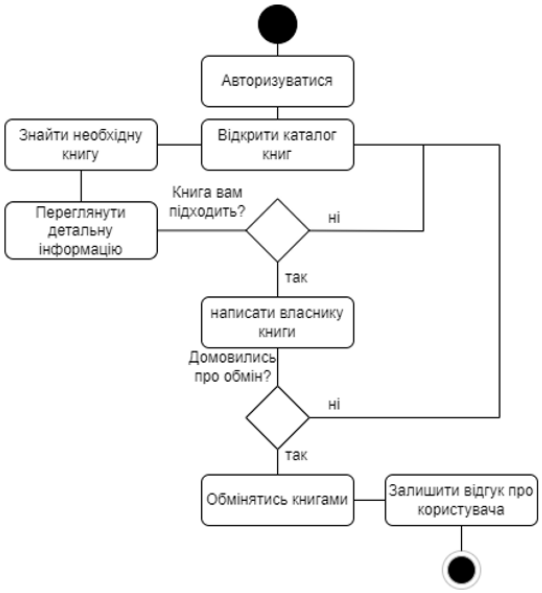
React

6

USE CASE ДІАГРАМА

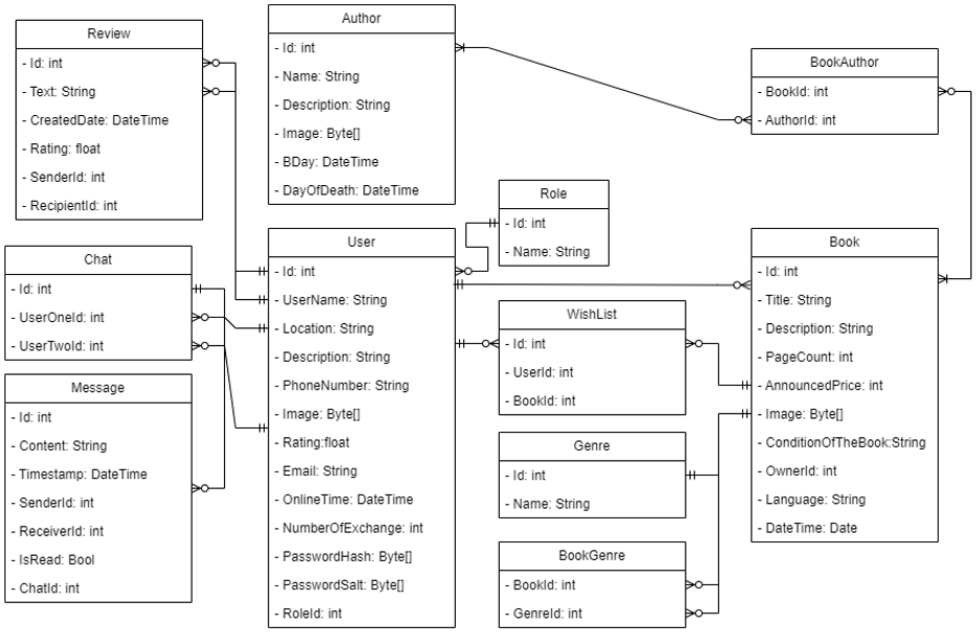
7

Діаграма послідовності обміну книги



8

ДІАГРАМА БАЗИ ДАНИХ

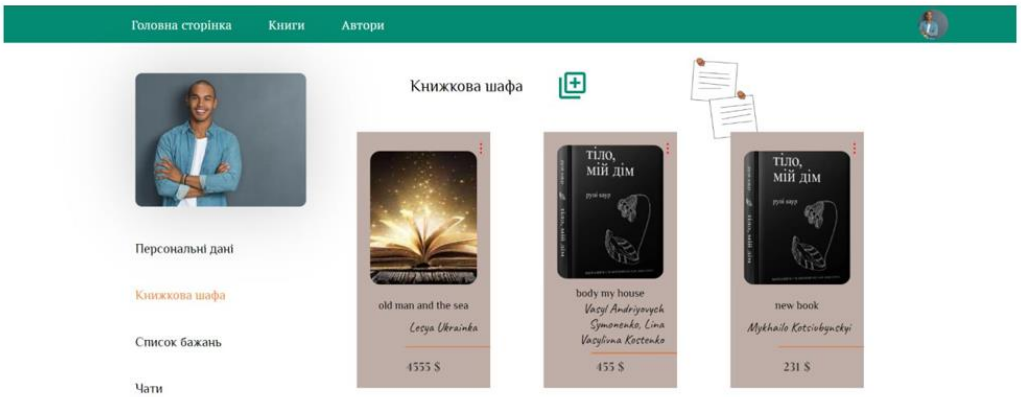


9

Мапа сайту

10

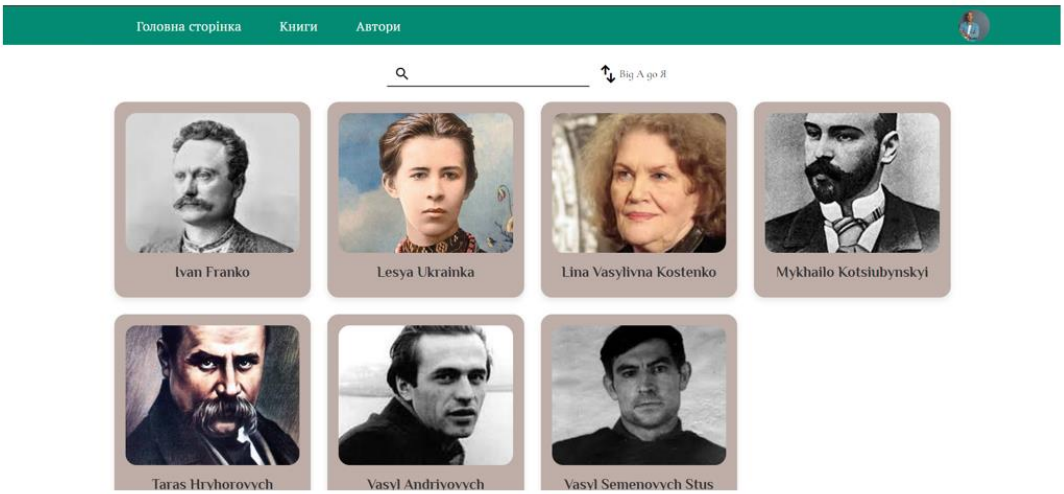
ЕКРАННІ ФОРМИ



Особиста книжкова шафа користувача

11

ЕКРАННІ ФОРМИ



Перегляд авторів

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Скворцов М.О., Оцінка ефективності баз даних для зберігання інформації про книги та користувачів у веб-додатку / Скворцов М.О, Замрій І.В. // Застосування програмного забезпечення в інформаційно-комунікаційних технологіях: Матеріали всеукраїнської науково-технічної конференції. 24 квітня 2024. Збірник тез К.: ДУІКТ, 2024. – С. 112
2. Скворцов М.О., Доповнена реальність як інструмент покращення навчальних процесів / Скворцов М.О, Замрій І.В. // Сучасні інтелектуальні інформаційні технології в науці та освіті: Матеріали четвертої всеукраїнської науково-технічної конференції. Подано до друку.

13

ВИСНОВКИ

1. Проведено огляд існуючих рішень та аналіз ринкових рішень показав, що багато існуючих платформ мають обмеження щодо функціональності, як от можливість продажу книги чи безпосередній обмін повідомленнями між користувачами.
2. Спроектовано архітектуру системи для веб-застосунку обміну друкованими книгами.
3. Спроектовано та розроблено базу даних системи обміну друкованими книгами.
4. Створено інтерфейс користувача системи обміну друкованими книгами.
5. Реалізовано основні функції веб-додатку, дотримуючись функціональних та нефункціональних вимог, таких як створення та керування особистим профілем, керування власними книгами та перегляд книг інших користувачів, перегляд авторів та створення необхідного автора, чат, список вподобаних книг, а також забезпечено захист від несанкціонованого доступу до даних користувача шляхом шифрування паролю користувача.
6. Застосунок проаналізовано за рахунок мануального тестування. В результаті тестування було виявлено значну затримку у відображенні книг, а саме 2.5 секунди, що вдалось вчасно виявити та знизити швидкість завантаження до 1 секунди.

14

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

Books.cs

```
using System.Text.Json.Serialization;

namespace Database.Models
{
   public class Book
   {
      public int Id { get; set; }
      public string? Title { get; set; }
      public string? Description { get; set; }
      public int PageCount { get; set; }
      public decimal AnnouncedPrice { get; set; }
      public byte[]? Image { get; set; }
      public string ConditionOfTheBook { get; set; }
      public int OwnerId { get; set; }
      public string? Language { get; set; }
      public DateTime DateTime { get; set; } = DateTime.Now;
      public virtual User? Owner { get; set; }
      [JsonIgnore]
      public virtual ICollection<BookAuthor> Authors { get; set;} = new List<BookAuthor>();
      [JsonIgnore]
      public virtual ICollection<BookGenre> Genres { get; set; } = new List<BookGenre>();
      public virtual ICollection<Wishlist> Wishlists { get; set; } = new List<Wishlist>();

   }
}
```

User.cs

```
namespace Database.Models
{
   public class User
   {
      public int Id { get; set; }
      public string? UserName { get; set; } = string.Empty;
      public string? Location { get; set; } = string.Empty;
      public string? Description { get; set; } = string.Empty;
      public string? PhoneNumber { get; set; }
      public float Rating { get; set; } = 0;
      public int NumberOfExchanges { get; set; } = 0;
      public DateTime OnlineTime { get; set; } = DateTime.Now;
      public string? Email { get; set; }
      public byte[]? PasswordHash { get; set; }
      public byte[]? PasswordSalt { get; set; }
      public byte[]? Image { get; set; }
      public Role? Role { get; set; }
      public int RoleId { get; set; }
      public virtual ICollection<Book> Books { get; set; } = new List<Book>();
      public virtual ICollection<UserReview> SentReviews { get; set; } = new List<UserReview>();
      public virtual ICollection<UserReview> ReceivedReviews { get; set; } = new List<UserReview>();
      public virtual ICollection<Wishlist> Wishlists { get; set; } = new List<Wishlist>();
      public virtual ICollection<Chat> Chats { get; set; } = new List<Chat>();

   }
}
```

BookController.cs

```
using BusinessLogic.ModelsDTO.BookDTO;
using BusinessLogic.Services.BookService;
using Microsoft.AspNetCore.Authorization;
```

```
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace ChangeBooksProject.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    public class BookController : ControllerBase
    {
        private readonly IBookService _bookService;

        public BookController(IBookService bookService)
        {
            _bookService = bookService;
        }
        [HttpGet("{id:int}")]
        public async Task<IActionResult> GetBookById(int id)
        {
            var book = await _bookService.GetBookById(id);

            if (book == null)
            {
                return NotFound();
            }

            return Ok(book);
        }

        [HttpPost("create")]
        public async Task<IActionResult> CreateBook([FromBody] CreateBookDTO bookDto)
        {
            var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
            if (userIdClaim == null)
            {
                return Unauthorized("User ID is missing from the token claims.");
            }
            var userId = int.Parse(userIdClaim.Value);
            var book = await _bookService.CreateBook(bookDto, userId);

            if (book == null)
            {
                return BadRequest("Error creating book");
            }

            return Ok(book);
        }

        [HttpGet("myBooks")]
        public async Task<IActionResult> GetMyBooks()
        {
            var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
            if (userIdClaim == null)
            {
                return Unauthorized("User ID is missing from the token claims.");
            }
            var userId = int.Parse(userIdClaim.Value);
            var books = await _bookService.GetMyBooks(userId);

            if (books == null || !books.Any())
            {
                return NotFound("No books found for the user.");
```

```
            }

            return Ok(books);
        }
        [HttpGet("search")]
        public async Task<IActionResult> SearchBooksByTitle(string query)
        {
            var books = await _bookService.SearchBooksByTitle(query);

            if (!books.Any())
            {
                return NotFound("No books found with the given title fragment.");
            }

            return Ok(books);
        }

        [HttpGet]
        public async Task<IActionResult> GetBooks([FromQuery] int pageNumber, [FromQuery] int pageSize, [FromQuery]
string? genreIds)
        {
            List<int> genreIdList = null;
            if (!string.IsNullOrEmpty(genreIds))
            {
                genreIdList = genreIds.Split(',').Select(int.Parse).ToList();
            }

            var books = await _bookService.GetBooks(pageNumber, pageSize, genreIdList);
            if (books == null || !books.Any())
            {
                return NotFound("No books found for the given filters.");
            }
            return Ok(books);
        }
        [HttpDelete("delete/{bookId:int}")]
        public async Task<IActionResult> DeleteBook(int bookId)
        {
            var book = await _bookService.GetBookById(bookId);
            if(book == null)
            {
                return NotFound("Book not found");
            }
            try
            {
                await _bookService.DeleteBook(bookId);
                return Ok($"Book has been successfully deleted.");
            }
            catch (Exception ex)
            {
                return BadRequest($"An error occurred while deleting the book: {ex.Message}");
            }
        }
    }
}
```

UserController.cs

```
using BusinessLogic.Services.UserService;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;
using BusinessLogic.ModelsDTO.UserDTO;
```

```
namespace ProjectWithMyLove.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UserController : ControllerBase
    {
        private readonly IUserService _userService;

        public UserController(IUserService userService)
        {
            this._userService = userService;
        }
        [HttpGet("{id:int}")]
        public async Task<IActionResult> GetById(int id)
        {
            var user = await _userService.UserGetById(id);

            if (user == null)
            {
                return NotFound();
            }

            return Ok(user);
        }

        [HttpPut("update")]
        public async Task<IActionResult> UpdateUser([FromBody] AddUserInfoDTO updateUserDTO)
        {
            var userIdClaim = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier)?.Value;

            if (userIdClaim == null)
            {
                return Unauthorized("User ID is missing from the token claims.");
            }

            if (!int.TryParse(userIdClaim, out var userId))
            {
                return BadRequest("Invalid user ID claim.");
            }

            var result = await _userService.UpdateUser(userId, updateUserDTO);
            return result ? Ok() : BadRequest("Can't update user.");
        }
        [HttpPost("changePassword")]
        public async Task<IActionResult> ChangePassword(ChangePasswordDTO changePasswordDTO)
        {
            var userIdClaim = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier)?.Value;

            if (userIdClaim == null)
            {
                return Unauthorized("User ID is missing from the token claims.");
            }

            if (!int.TryParse(userIdClaim, out var userId))
            {
                return BadRequest("Invalid user ID claim.");
            }

            var result = await _userService.ChangePassword(userId, changePasswordDTO.CurrentPassword,
changePasswordDTO.NewPassword);
            if (result)
            {
```

```
            return Ok("Password changed successfully.");
        }
        else
        {
            return BadRequest("Failed to change password.");
        }
    }
    [HttpDelete("delete/{id}")]
    public async Task<IActionResult> DeleteUser(int id)
    {
        var result = await _userService.DeleteUser(id);
        if (result)
        {
            return Ok();
        }
        else
        {
            return NotFound();
        }
    }

  }
}
```

AuthorController.cs

```
using BusinessLogic.ModelsDTO.AuthorDTO;
using BusinessLogic.ModelsDTO.UserDTO;
using BusinessLogic.Services.AuthorServcie;
using Database.Models;
using Database.Repositories.Implements;
using Microsoft.AspNetCore.Mvc;

namespace ChangeBooksProject.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class AuthorController : ControllerBase
  {
    private readonly IAuthorService _authorService;

    public AuthorController(IAuthorService authorService)
    {
        _authorService = authorService;
    }
    [HttpGet("{id:int}")]
    public async Task<IActionResult> GetById(int id)
    {
        var author = await _authorService.AuthorGetById(id);

        if (author == null)
        {
            return NotFound();
        }

        return Ok(author);
    }

    [HttpPost("create")]
    public async Task<IActionResult> CreateAuthor([FromBody] CreateAuthorDTO authorDto)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
```

```
            }

            var author = await _authorService.CreateAuthor(authorDto);

            if (author == null)
            {
                return BadRequest("Error creating author");
            }

            return Ok(author);
        }
        [HttpGet("all")]
        public async Task<IActionResult> GetAllAuthor()
        {
            var author = await _authorService.GetAllAuthor();

            if (author == null)
            {
                return NotFound();
            }

            return Ok(author);
        }
        [HttpGet("paginated")]
        public async Task<IActionResult> GetAuthorsPaginated(int pageNumber = 1, int pageSize = 8)
        {
            var (authors, totalAuthors) = await _authorService.GetAuthorsPaginated(pageNumber, pageSize);

            if (!authors.Any())
            {
                return NotFound();
            }

            var response = new
            {
                TotalAuthors = totalAuthors,
                Authors = authors,
                PageNumber = pageNumber,
                PageSize = pageSize
            };

            return Ok(response);
        }
    }
}
```

UserProfile.tsx

```
import React, { useState, useEffect, useContext } from "react";
import Sidebar from "./Sidebar";
import "../styles/userProfile.css";
import { useUserContext } from "../contexts/UserContext";
import Button from "./Button";
import axios from "../utils/axios";
import SuccessMessage from "./SuccessMessage";
import { useNavigate } from "react-router-dom";

const UserProfile: React.FC = () => {
  const { userId, user, setUser, image, profileDeleted, setProfileDeleted } =
    useUserContext();
  const navigate = useNavigate();
  const [successInfo, setSuccessInfo] = useState({
    isVisible: false,
```

```
  message: "",
  imageSrc: "",
  type: "",
});

useEffect(() => {
  const fetchUser = async () => {
    try {
      const response = await axios.get(`/User/${userId}`, {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      });
      setUser(response.data);
    } catch (error) {
      console.error("An unexpected error occurred", error);
    }
  };

  fetchUser();
}, [userId, profileDeleted, navigate, setProfileDeleted, setUser]);

const handleSave = async () => {
  if (!user) {
    console.error("No user data to save");
    return;
  }

  try {
    const config = {
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${localStorage.getItem("token")}`,
      },
    };

    const updateUserInfo = {
      UserName: user.userName,
      Description: user.description,
      Image: image,
      Location: user.location,
      PhoneNumber: user.phoneNumber,
      Email: user.email,
    };
    const body = JSON.stringify(updateUserInfo);
    const response = await axios.put("/User/update", body, config);

    setSuccessInfo({
      isVisible: true,
      message: "Profile updated successfully!",
      imageSrc: "/images/success-icon.png",
      type: "update-success",
    });
  } catch (error) {
    console.error("An error occurred while updating the user", error);
    alert("Failed to update profile. Please try again.");
  }
};

return (
  <>
    <div className="user-profile">
```

```
    <Sidebar />
    <div className="user-details">
      <h2 className="personal-data">Персональні дані</h2>
      <img src="/images/flyingBooks.png" className="flying-books"></img>
      <p className="user-profile-p">Ім'я</p>
      <input
        type="text"
        value={user ? user.userName : ""}
        onChange={(e) =>
          user && setUser({ ...user, userName: e.target.value })
        }
      />
      <p className="user-profile-p">Номер телефону</p>
      <input
        type="text"
        value={user ? user.phoneNumber : ""}
        onChange={(e) =>
          user && setUser({ ...user, phoneNumber: e.target.value })
        }
      />
      <p className="user-profile-p">Пошта</p>
      <input
        type="text"
        value={user ? user.email : ""}
        onChange={(e) =>
          user && setUser({ ...user, email: e.target.value })
        }
      />
      <p className="user-profile-p">Місце знаходження</p>
      <input
        type="text"
        value={user ? user.location : ""}
        onChange={(e) =>
          user && setUser({ ...user, location: e.target.value })
        }
      />
      <p className="user-profile-p">Про мене</p>
      <textarea
        value={user ? user.description : ""}
        onChange={(e) =>
          user && setUser({ ...user, description: e.target.value })
        }
      />
      <Button onClick={handleSave} className="user-profile-save-button">
        Зберегти
      </Button>
    </div>
  </div>
  <img src="/images/ManReading.png" className="man-reading"></img>
  <SuccessMessage
    isVisible={successInfo.isVisible}
    onClose={() => {
      setSuccessInfo({ ...successInfo, isVisible: false });
      if (successInfo.type === "delete-success") {
        navigate("/login");
      }
    }}
    message={successInfo.message}
    imageSrc={successInfo.imageSrc}
    type={successInfo.type}
  />
</>
```

```
  );
};

export default UserProfile;
BookDetails.tsx
import React, { useState, useEffect } from "react";
import { useParams } from "react-router-dom";
import axios from "../utils/axios";
import { Book } from "../types/BookTypes";
import "../styles/bookDetails.css";

const BookDetails = () => {
  const { bookId } = useParams();
  const [book, setBook] = useState<Book | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");

  useEffect(() => {
    const fetchBookDetails = async () => {
      try {
        const response = await axios.get(`/book/${bookId}`);
        setBook(response.data);
      } catch (err) {
        setError("Failed to fetch book details. Please try again later.");
      } finally {
        setLoading(false);
      }
    };

    fetchBookDetails();
  }, [bookId]);

  if (loading) return <div>Loading...</div>;
  if (error) return <div>{error}</div>;
  if (!book) return <div>No book found.</div>;

  return (
    <div className="all-book-container">
      <div className="top-book">
        <div className="top-book-image">
          <img
            src={
              book.image
                ? `${book.image}`
                : "/images/noImage.png"
            }
            alt={book.title}
            className="book-details-image"
          />
        </div>
        <div className="top-book-right">
          <h1 className="book-details-title">{book.title}</h1>
          <div className="book-details-author">
            {book.authors?.map((author) => author.name).join(", ")}
          </div>
          <div className="book-details-genres">
            {book.genres?.map((genre) => (
              <span className="genre-badge">{genre.name}</span>
            ))}
          </div>

          <div className="book-details-specs">
```

```
          <span className="book-info">Language: {book.language}</span>
          <span className="book-info">{book.pageCount} pages</span>
          <span className="book-info">{book.conditionOfTheBook}</span>
        </div>
        <p className="book-details-description">{book.description}</p>
      </div>
    </div>
    <div className="bottom">
      <div className="bottom-left">
        <img
          src={
            book.owner?.image
              ? `${book.owner.image}`
              : "/images/default-user.png"
          }
          alt="Owner"
          className="book-details-owner-image"
        />
        <div className="book-details-owner-details">
          <p className="book-details-owner-name">{book.owner?.userName}</p>
          <p className="owner-last-visit">
            Last visit:{" "}
            {book.owner?.onlineTime
              ? new Date(book.owner.onlineTime).toLocaleDateString("uk-UA", {
                  day: "2-digit",
                  month: "2-digit",
                })
              : "Late"}
          </p>
        </div>
      </div>
      <div className="book-details-owner-rating">
        {book.owner?.rating?.toFixed(1) ?? "N/A"}
        <span
          className="star-icon"
          style={{
            backgroundImage: `linear-gradient(to right, #ffac33 ${
              book.owner?.rating ? (book.owner.rating / 5) * 100 : 0
            }%, grey ${
              book.owner?.rating ? (book.owner.rating / 5) * 100 : 0
            }%)`,
          }}
        >
          ★
        </span>
      </div>
      <span className="book-info price"> {book.announcedPrice} €</span>
      <div className="bottom-right">
        <button
          className="book-details-button button"
          onClick={() => alert("Message to owner")}
        >
          Write to the owner
        </button>
      </div>
    </div>
    <img className="woman-reading" src="/images/Woman-reading.png" />
    <hr />
  </div>
 );
};
export default BookDetails;
```