МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Центр заочної, дистанційної та вечірньої форм навчання

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

11 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 – Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна технологія проектування системи онлайн буккросингу»

здобувачки групи Ін.мз-21с Дідик Віталіни Іванівни

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

_____ Віталіна ДІДИК

(підпис)

Керівник

доцентка кафедри комп'ютерних наук,

к.т.н., доцент Наталія БАРЧЕНКО _____

(підпис)

Суми – 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувачки групи ІН.мз-21с

1. Тема роботи: «Інформаційна технологія проектування системи онлайн буккросингу» затверджую наказом по СумДУ від *«20» листопада 2023 р. № 1308-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 13 грудня 2023 року*
3. Вхідні дані до кваліфікаційної роботи

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Опис предметної області, обґрунтування оптимального варіанту реалізації мети цієї роботи. 2) Вибір програмних засобів реалізації. 3) Програмна реалізація інформаційної веб-системи для онлайн обміну книгами.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20__ р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми та актуальності розробки системи для обміну книжками онлайн, постановка й формування завдань дослідження*	14.10.2023	Виконано
2	*Огляд та вибір програмних засобів для інформаційної системи.*	25.10.2023	Виконано
3	*Розроблення інформаційної системи онлайн буккросингу*	15.11.2023	Виконано
4	*Аналіз отриманих результатів*	22.11.2013	Виконано
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	01.12.2023	Виконано

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 78 стор., 23 рис., 1 додаток, 29 використаних джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки присвячена популяризації літературної індустрії шляхом розробки інформаційної технології для онлайн обміну книгами.

Об'єкт дослідження — онлайн буккросинг.

Мета роботи — розробка та вдосконалення інформаційної технології для обміну книгами онлайн з метою задоволення потреб читацької спільноти та використання сучасних технологічних можливостей.

Методи дослідження — мови програмування PHP (фреймворк Laravel), JavaScript (фреймворк vue.js).

Результати — розроблено інформаційну систему онлайн буккросинг, надає змогу користувачу реєструватись, авторизуватись, редагувати дані в особистому кабінеті, додавати книги та обмінюватись з іншими користувачами, розроблена функція внутрішнього чату для діалогу користуваців.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ЛОГІЧНА СХЕМА, ДІАГРАМА ВИКОРИСТАННЯ, PHP, MYSQL, LARAVEL, VUE.JS.

ЗМІСТ

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки з кожним роком підвищується попит на доступ до книг через онлайн-платформи. Зокрема, пандемія та інші обставини, які обмежують користувачів до доступу бібліотек та книгарень. Також, зміни в уподобаннях читачів підкреслюють необхідність створення сучасних інформаційних систем для обміну книгами онлайн.

Об’єкт дослідження. Інформаційна система для обміну книжками.

Предмет дослідження. Розробка та вдосконалення інформаційної системи для онлайн обміну книгами з метою задоволення потреб читацької спільноти та використання сучасних технологічних можливостей.

Гіпотеза. Розробка та впровадження сучасної інформаційної системи для обміну книгами онлайн, яка відповідає сучасним вимогам щодо дизайну та функціональності, сприятиме полегшенню доступу до літератури, популяризації літературної індустрії та сприятиме збереженню довкілля, завдяки багаторазовому використанню книг.

Новизна. Описане у даній роботі програмне рішення дозволить досягти більшої зручної функціональності та відповідності вимогам користувачів. Інформаційна система на основі історій замовлень рекомендуватиме книги користувачу відповідно до обраних жанрів, що допоможе підвищити популяризацію до читання.

Структура. Дана робота складається зі вступу, інформаційного огляду, вибір засобів та мов програмування для реалізації поставленої задачі, опису практичної реалізації проекту, висновків, списку використаних джерел та додатків.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

У сучасному інформаційному суспільстві зростає попит на доступ до книжкової літератури через онлайн-платформи, що створює потребу в розвитку та вдосконаленні веб-систем для обміну книгами. Проте існуючі платформи не завжди задовольняють потреби користувачів та не враховують сучасні технологічні можливості. Отже, головною проблемою, яку необхідно вирішити, є розробка та впровадження оптимізованої веб-системи для обміну книгами, яка задовольнятиме потреби користувачів та використовуватиме сучасні технологічні рішення. Дослідження цієї проблеми дозволить вдосконалити існуючі рішення та розробити новий підхід до обміну книжками через Інтернет.

«Історія буккросингу почалася у травні 2001 року. Тоді американець Рон Горнбекер, спеціаліст у сфері Інтернет-технологій, залишив в холі готелю 20 книжок з пояснювальними написами. Через півроку на його сайті було вже 300 активних користувачів, які "відпускали" книги і приводили нових учасників. Рух із США поширився в Європу. Не оминув цей рух і України. Загалом зараз у мережі розповсюдження книг зареєстровано понад 250 тис. людей і більше мільйона книжок» [1].

Концепція буккросингу — це інноваційний метод популяризації книг та читання, що базується на ідеї передачі книг безкоштовно через метод "обміну".

Буккросинг — це практика обміну книгами, залишаючи їх в громадських місцях або на спеціально відведених полицях. Ця ідея передбачає можливість для шукачів взяти та прочитати ці книги, а також розповсюджувати їх подальше. Буккросинг ґрунтується на переконанні, що література повинна подорожувати та обмінюватися, живлячи та вдосконалюючи себе через читання.

Буккросинг об'єднує мільйони любителів читання по всьому світу, особливо тих, хто активно використовує інноваційні технології.

На сьогоднішній день існує безліч громацьких місць для буккросингу, де книги можна залишити або взяти у борг. Зазвичай вони облаштовані в різноманітних локаціях, таких як бібліотеки, культурно-освітні заклади, молодіжні клуби, кафе, торгові центри або в офісі на роботі.

Проте фізичний буккросинг може бути неактуальним в умовах, коли доступ до громадських місць обмежений або коли існують інші обмеження, які вимагають від людей залишатися вдома. Однак існує можливість адаптувати ідею буккросингу до онлайн-середовища, щоб забезпечити доступ до літератури та популяризувати читання навіть в умовах ізоляції.

Мета створення ресурсу для користувачів в Україні полягає в полегшенні доступу до книжкової літератури та стимулюванні читацької активності серед українського населення. Цей ресурс призначений для того, щоб надати українцям легко та зручно обмінюватися книжками з іншими користувачами, без обмежень за місцем проживання чи соціальним статусом. Це сприятиме підвищенню читацької активності в Україні, розвитку культури читання та розповсюдженню літературних творів серед української аудиторії.

1.2 Огляд існуючих рішень

На сучасному ринку існує ряд платформ для буккросингу та обміну книгами, які надають користувачам можливість ділитися та отримувати книги. Серед таких платформ можна виділити "BookCrossing.com", "Goodreads", "BookMooch" та інші.

BookCrossing є однією з найвідоміших та найпоширеніших платформ для фізичного буккросингу. Користувачі можуть залишати книги в громадських місцях, позначаючи їх спеціальними етикетками з унікальними ідентифікаторами. Інші читачі можуть знаходити ці книги, читати їх, реєструвати їхні подальші переміщення та ділитися враженнями на веб-сайті BookCrossing. Ця платформа створює велику спільноту читачів та дозволяє книгам подорожувати світом [2].

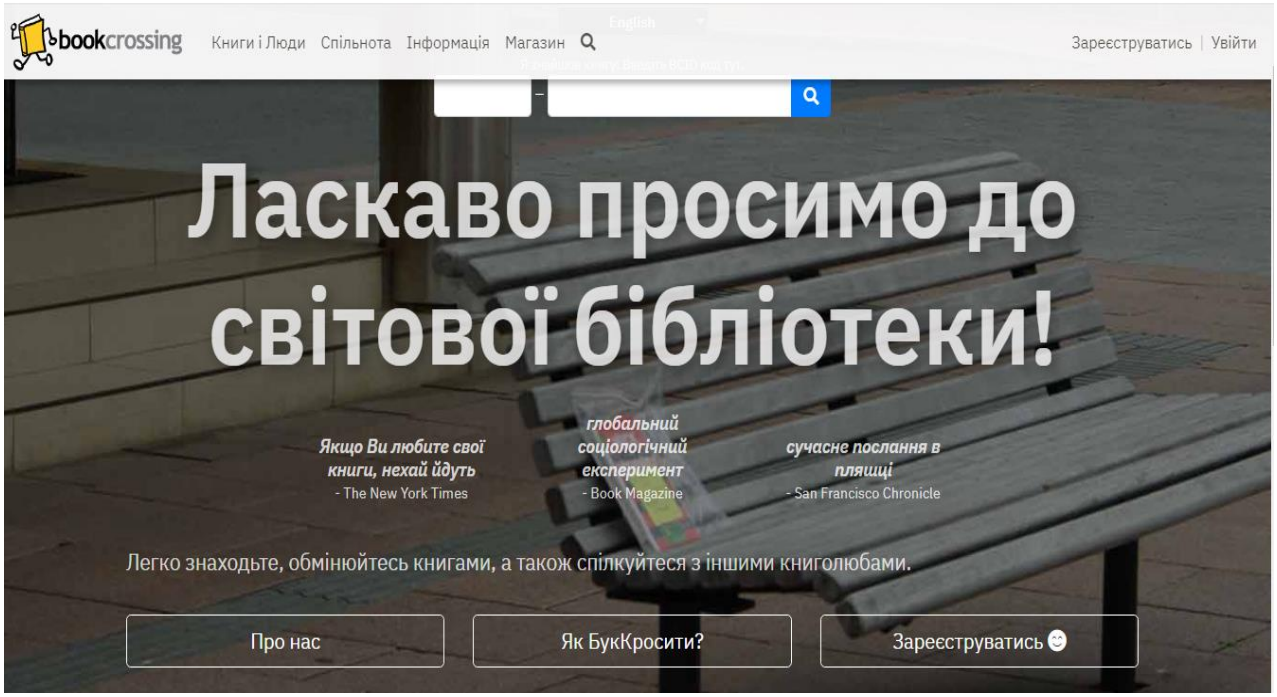

Рисунок 1.1 – Інтерфейс сайту «BookCrossing»

Переваги даної веб-системи:

- BookCrossing має велику та активну спільноту читачів по всьому світу. Це дозволяє книгам подорожувати світом і забезпечує можливість знаходити книги, які інакше було б важко знайти.
- Користувачі можуть легко долучитися до BookCrossing, залишаючи книги в громадських місцях та позначаючи їх унікальними етикетками. Онлайн-реєстрація книг також дуже проста.
- BookCrossing надає можливість користувачам залишати рецензії та коментарі про книги, які вони прочитали, та ділитися своїми враженнями з іншими читачами.
- BookCrossing стимулює створення та підтримку локальних спільнот буккросерів, які організовують зустрічі та події для обміну книгами.

Недоліки платформи BookCrossing:

- Фізичні обмеження, спеціалізується на фізичному обміні книгами, тому він не надає можливості обмінюватися електронними версіями книг або аудіокнигами.

- Іноді книги, залишені в громадських місцях, можуть бути втрачені або викрадені, що призводить до втрати літератури та ресурсів користувачів.
- BookCrossing не надає контролю над тим, хто бере книги та що з ними робить після того, як вони були залишені. Це може призвести до незадовільного вигляду книг або їх втрати.

Goodreads - це соціальна мережа для читачів, де користувачі можуть створювати власні профілі, відзначати прочитані книги, ставити їм оцінки та залишати рецензії. Платформа також надає можливість створювати список бажань та обмінюватися рекомендаціями з іншими читачами. Goodreads дозволяє створювати власні читацькі групи та приєднуватися до існуючих, де можна обговорювати книги та організовувати читацькі виклики [3].

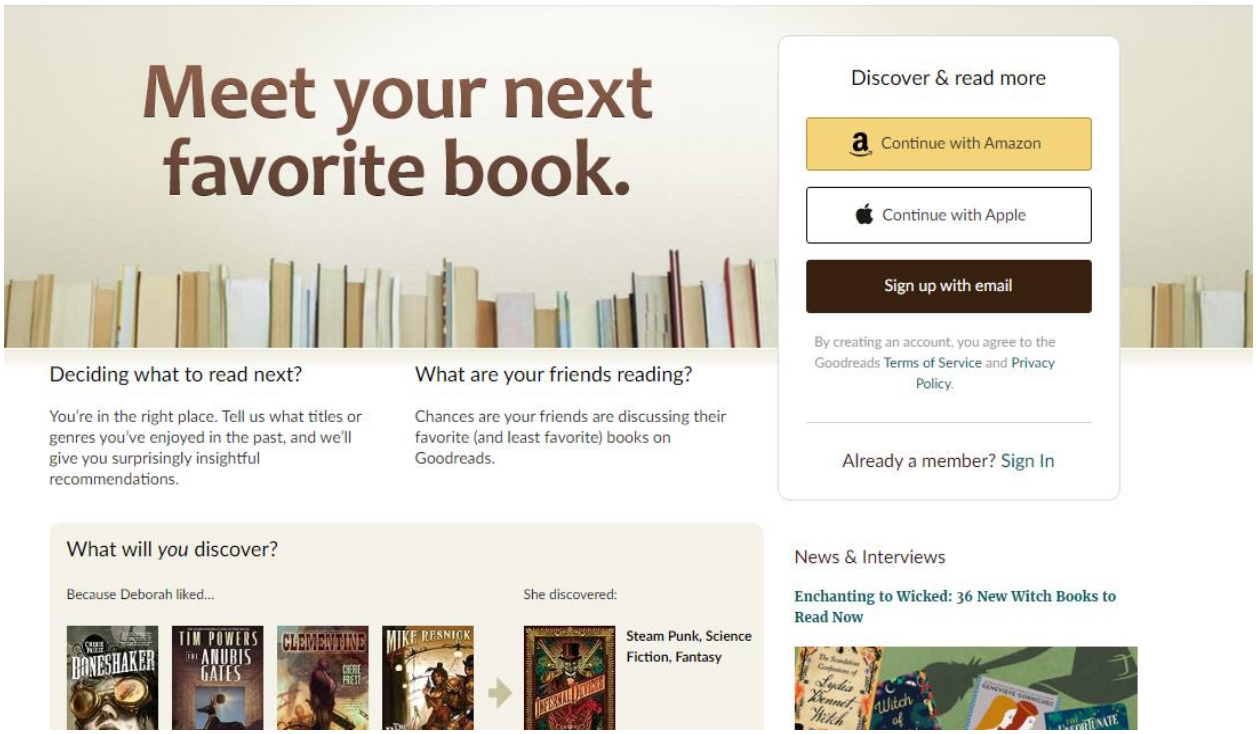

Рисунок 1.2 – Інтерфейс сайту «Goodreads»

Переваги цього сервісу:

- Goodreads має активну та велику спільноту користувачів, що створює можливість знайти читачів зі схожими інтересами,

обговорювати книги та долучитися до читацьких груп.

- Користувачі можуть створювати власні віртуальні бібліотеки, відзначати книги, які вони прочитали, та створювати список книг, які їх цікавлять. Це допомагає відстежувати свій читацький прогрес і спрощує організацію своєї особистої бібліотеки.
- Користувачі можуть залишати рецензії та оцінки для прочитаних книг, що допомагає іншим читачам вибирати книги для читання та визначати їхню якість.

Недоліки платформи Goodreads:

- Деякі користувачі можуть вважати, що Goodreads має обмежену функціональність порівняно з іншими соціальними мережами або читацькими платформами.
- Локазізація, хоч і технологічно в Україні можна використовувати цей ресурс, але логістично не зрозуміло як користувач може отрмати книгу та відправити свою.
- Goodreads не надає можливість завантажувати або обмінюватися електронними версіями книг, що може бути недоліком для тих, хто бажає читати в електронному форматі.
- Платформа Goodreads належить Amazon, що може створювати конфлікти інтересів і обмежувати доступ до книг та послуг для користувачів інших онлайн-книжкових магазинів.

Books Around - це додаток для організації та каталогізації власної бібліотеки. Користувачі можуть завантажувати свої книги, обмінювати та продавати. Books Around також надає можливість використовувати інтуруктивну карту для локалізаці пошуку книг [4].

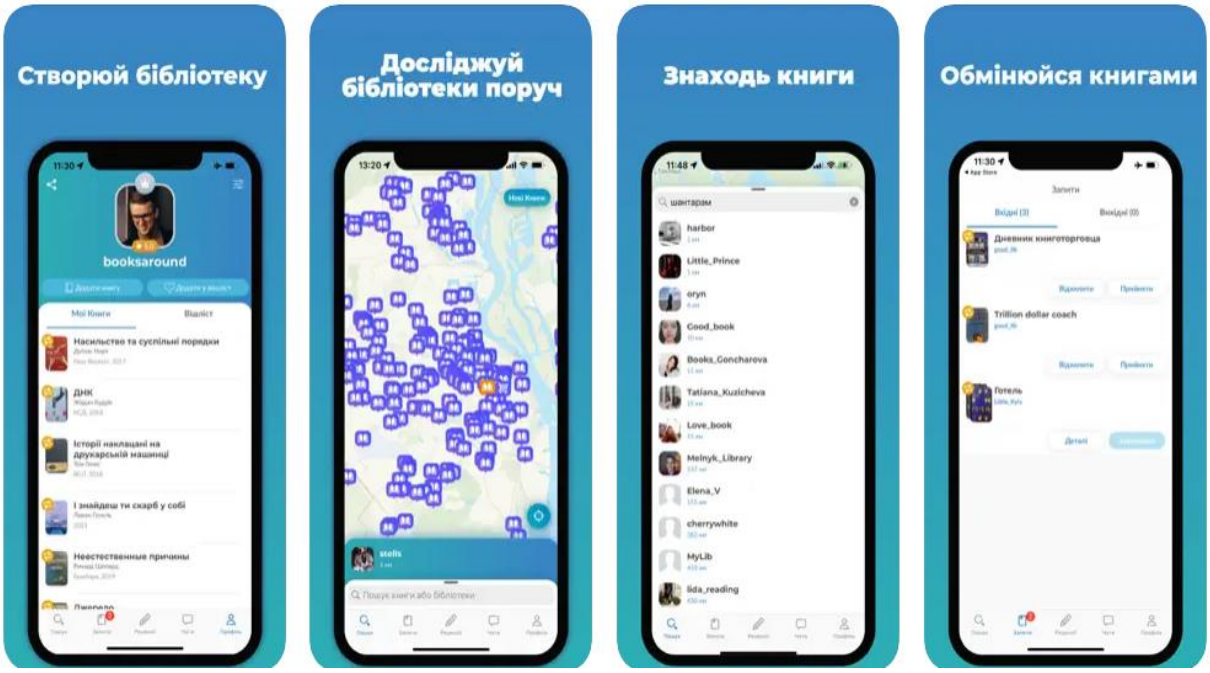

Рисунок 1.3 – Інтерфейс сайту «Books Around»

Переваги цього додатку:

- Додаток доступний для користувачів в Україні і має інтерактивну карту, що показує місця, де знаходяться книги. Це сприяє зручному пошуку книг у певному регіоні.
- Вимагає реєстрацію, що допомагає підвищити рівень безпеки та відслідковувати активність користувачів.
- Користувачі можуть придбати підписку "Книгарня" або "Автор", що розширює їхні права та можливості використання додатку.
- Присутність функції чату дозволяє користувачам спілкуватися один з одним, обмінюватися інформацією та обговорювати обмін книгами.

Серез недоліків цього додатку:

- Відсутність веб-версії додатку може ускладнити користувачам доступ до платформи через браузер на комп'ютері або інших пристроях.
- В інтерфейсі не завжди зрозуміло, що для пошуку книг можна

використовувати функцію "Пошук", оскільки увага користувачів привертається інтерактивною картою. Це може призвести до заплутання користувачів.

- Вимагає обов'язкової реєстрації, навіть для перегляду каталогу з наявністю книг. Таким чином, користувачам потрібно надавати свою інформацію, навіть якщо вони просто хочуть подивитися, які книги доступні.
- Відсутність можливості вийти з облікового запису користувача без його видалення може створювати незручності, особливо якщо користувач хоче змінити або відновити свій обліковий запис.

Загалом, додаток "Books Around" має свої переваги, такі як локалізація та захист, але також має обмеження та недоліки, які можуть вплинути на зручність користування та задоволення від використання платформи.

Також, існує обмін книгами за допомогою соціальних мереж або месенджерів. Це досить популярний метод обміну книжками [5].

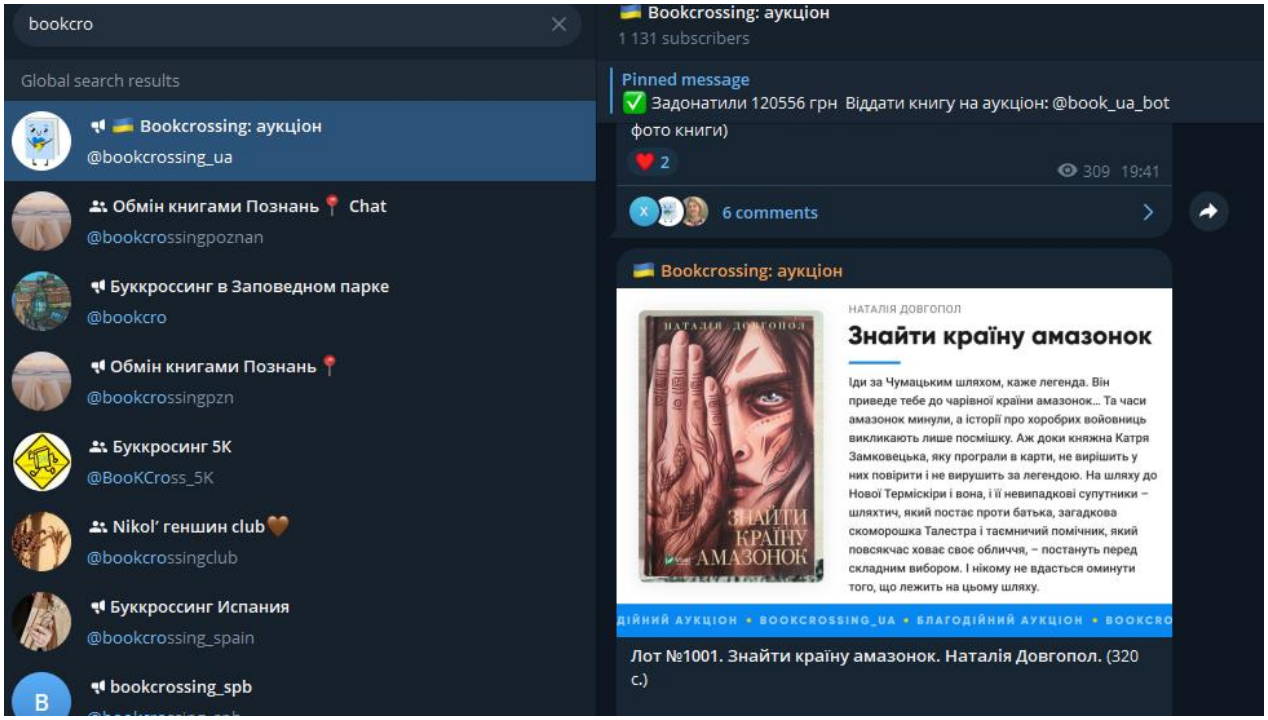

Рисунок 1.4 – Онлайн буккросинг через месенджери

Перевагою такого підходу є - зручність та швидкість. Соціальні мережі та месенджери забезпечують швидкий та безпосередній спосіб зв'язку між користувачами, що дозволяє швидко домовлятися про обмін та встановлювати деталі.

Проте особливу увагу хотілось звернути на недоліки. В цьому аспекті важливим фактором є приватність і безпека. Обмін даними через соціальні мережі або месенджери може вимагати розкриття особистих даних, що може бути проблематичним для деяких користувачів. Обмін через соціальні мережі може бути менш організованим і структурованим порівняно з додатками або платформами, спеціально розробленими для обміну книгами. Таким чином, користувачі можуть стати жертвами шахрайства та отримати послуги, на які вони розраховували, використовуючи цей спосіб обміну. Та останок, знайти конкретну книгу або відстежити історію обміну може бути більш складною через соціальні мережі або месенджери.

1.3 Дослідження актуальності проблеми

На основі аналізу існуючих реалізованих рішень, можу виділити основні проблеми використання уже створених ресурсів для обміну книжками:

1. На Українському ринку майже відсутні сервіси для обміну книжками, що обмежує можливості розвитку літератури та популяризації читання.

2. Відсутні веб системи, які задовольняли б користувацькі потреби. Інтерфейси не відповідає сучасним очікуванням користувачів. У світі, де технології швидко розвиваються, люди очікують, що додатки будуть не лише ефективними, але й легкими в користуванні, зручними та привабливими. З цим розвитком суспільства важливість дизайну та взаємодії з користувачем (UI/UX) стає ще більш актуальною [20].

3. Використання готових платформ, таких як соціальні мережі чи месенджери, може призвести до деяких проблем. Наприклад, це включає

високу залежність від розробників платформи та обмежену можливість налаштування та розвитку проекту [22].

Отже, аналізуючи наявні проблеми розробка нової веб-системи для обміну книгами є необхідним. Головна мета цього проекту полягає в полегшенні обміну книгами та поліпшенні дизайну додатку, щоб зробити його простим та привабливим для звичайних користувачів.

1.4 Постановка задачі

Основною метою інформаційної системи для буккросингу є створення системи, яка буде задовільняти потреби користувачів у використанні. Тож, веб система має бути зручною, мати зрозумілий та інтуїтивний інтерфейс, можливіть спілкування між користистувачами, мати можливість авторизації та можливіть редагування даних.

Для реалізаціїї всіх можливостей системи, складемо функціональні та не функціональні вимоги [26].

Основними функціональними вимогами є:

- Реєстрація та авторизація. Система повинна надавати можливість користувачам вводити основні дані у форму реєстраціїї. Для забезпечення безпеки та ідентифікації користувачі повинні мати можливість авторизуватися під своїм обліковим записом.
- Додавання книг. Система повинна мати функцію додавання книг користувачами та мати можливіть їх редагування.
- Користувачі повинні мати можливість ініціювати обмін книгами з іншими учасниками системи.
- Можливість підтвердити або відхилити запит на обмін книгами.
- Система повинна забезпечувати можливість обговорення деталей обміну через внутрішню систему повідомлень.
- Сповіщення та повідомлення. Система повинна підтримувати сповіщення та повідомлення для користувачів, щоб повідомляти їх про нові обміни, запити на обмін, а також інші важливі події.

- Профіль користувача. Кожен користувач повинен мати свій особистий профіль, де можна переглядати інформацію про обміни, додані книги, відгуки та рейтинг.

Не функціональними вимогами є:

- Система повинна мати високу продуктивність, надавати швидкий та ефективний доступ до інформації.
- Система повинна бути надійною та стійкою до відмов, забезпечуючи безперервну доступність сервісу для користувачів [27].
- Забезпечення конфіденційності та цілісності даних, використання надійних методів аутентифікації та авторизації.
- Система повинна бути сумісною з різними браузерами та пристроями, щоб користувачі мали можливість використовувати її на різних платформах.
- Інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним для користування.

Розробка такого додатку може задовольнити сучасні потреби читацької спільноти.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір фреймворків для реаліалізації веб системи

Ключовим етапом в розробці будь-якого веб-додатку є обрання підходящих засобів і технологій. У цьому розділі аналізуємо фреймворки для розробки веб-додатку для обміну книгами. Фреймворк - це ключовий інструмент для розробників, який спрощує створення високоякісних та функціональних веб-додатків. Від вибору інструментів залежить продуктивність, безпека, гнучкість та можливості майбутнього нашого сайту [6].

PHP має широкий вибір фреймворків, але не всі вони підходять під наші критерії. Тож, розглянемо: Codeigniter, Kohana, Laravel [9].

Laravel. Laravel представляє собою безкоштовний фреймворк з відкритим вихідним кодом, призначений для розробки веб-додатків за архітектурним шаблоном model–view–controller (MVC). Відзначається значною популярністю, Laravel включає модульну систему упакування з виокремленим менеджером залежностей, різні методи доступу до реляційних баз даних та корисні утиліти для розгортання і технічного обслуговування додатків. Laravel також надає можливості для роботи з базами даних, використовує сучасні практики [7].

Можемо виділити переваги цього фреймворку:

- зручність та простота у використанні;
- висока продуктивність, завдяки вбудованим інструментам, які допомагають оптимізувати завантаження сервера і прискорювати веб-додаток [21];
- має швидкий та ефективний шаблонний механізм, що допоможе у створені привабливих та функціональних сторінок;
- має вбудовані бібліотеки та модулі (аутентифікація, авторизація, кешування, маршрутизація);
- підтримує різні бази даних, такі як MySQL, PostgreSQL, MSSQL,

SQLite, Oracle, і має готові рішення для різноманітних завдань [8].

CodeIgniter. CodeIgniter - це фреймворк для розробки веб-систем та застосунків на мові програмування PHP. Він використовує модель-вид-контролер (MVC) та підтримує різні бази даних, має добре документовану архітектуру та готові рішення для різних завдань [18]. Головна ціль фреймворка CodeIgniter заключається в тому, щоб дозволити розробляти проекти набагато швидше, ніж якщо б ви писали весь код самі, представляючи величезний набір готових бібліотек для вирішення задач, які найчастіше зустрічаються, а також простий інтерфейс і логічну структуру для доступу до цих бібліотек. Видається досить простим у вивченні та розробці, і відзначається своєю продуктивністю [10].

Перевагами є:

- простота у вивчені;
- висока продуктивність;
- підтримує модульність, що дозволяє розділяти код на окремі модулі і бібліотеки [22].

Kohana. Kohana - це веб-фреймворк з відкритим кодом для розробки веб-систем на PHP, який використовує архітектурну модель MVC [20]. Він є безпечним, та простим у використанні. У порівнянні з іншими фреймворками, Kohana відзначається своєю легкістю та гнучкістю [11].

Даний інструмент має декілька основних переваг:

- високий рівень безпеки;
- архітектурна модель MVC, що дозволяє чітко розділити логіку додатку;
- легкість розширення;
- кешування.

Після уважного аналізу вищезазначених фреймворків для нашої веб-системи обрано Laravel в якості бекенд-інструменту. Вибір здійснений через такі ключові переваги:

- детальна документація;

- гнучкість;
- легкість в освоєнні;
- висока швидкість роботи;
- обширна бібліотека для вирішення різноманітних завдань. [23].

Обираючи цей фреймворк, отримаємо наступні переваги:

1. Однією з ключових переваг фреймворку Laravel є вбудований механізм роботи з об'єктно-реляційною картографією (ORM). ORM в Laravel дозволяє взаємодіяти з базою даних, використовуючи об'єктно-орієнтований підхід, замість традиційного SQL-коду [13].

2. Широкий функціонал та екосистема: Laravel має розгорнутий функціонал та широку екосистему додаткових пакетів та бібліотек, що дозволить легко розширювати функціонал сайту та використовувати готові рішення для таких завдань, як автентифікація, авторизація, робота з базою даних та багато інших [12].

3. Зручний синтаксис та швидкість розробки: Laravel відомий своїм зручним та елегантним синтаксисом, що робить розробку швидкою та приємною. Завдяки функціям, таким як готові шаблони та генерація коду, розробка нових функцій сайту відбувається швидше.

4. Активна спільнота та оновлення: Laravel має велику та активну спільноту розробників, яка постійно працює над покращеннями та оновленнями фреймворку. Це гарантує актуальність та безпеку веб-додатку [14].

5. Модульність та розширюваність: Laravel сприяє створенню модульної структури додатку, що спрощує його підтримку та розширення. Ви можете легко додавати новий функціонал, не змінюючи основний код [25].

6. Підтримка реалізації RESTful API - Laravel надає можливість легко розробляти RESTful API, що дозволить інтегрувати ваш веб-додаток з іншими сервісами та додатками. В майбутньому дозволить розширювати можливості веб системи для обміну книжками [24].

7. Швидкість та продуктивність: Laravel володіє високою швидкістю виконання завдяки використанню кешування, оптимізації запитів до бази даних та іншим технікам [15].

Laravel допоможе вам реалізувати всі необхідні функції та забезпечити комфортне використання сайту користувачами.

Для реалізації фронтенду веб системи буде використовуватись інструмент Vue.js. Vue.js - це прогресивний фреймворк для створення інтерфейсу користувача, який надає можливість реактивного програмування. Це означає, що зміни в даних автоматично відображаються в інтерфейсі, що робить користувацький досвід більш ефективним [16].

Цей інструмент має простий та зрозумілий синтаксис, що дозволяє освоїти фреймворк. Також, Vue.js добре інтегрується з іншими бібліотеками, що дозволяє розширювати його функціональність та використовувати його разом з іншими технологіями [17].

Обираючи Laravel для бекенду та Vue.js для фронтенду, ми можемо створити високоякісний веб-сайт для буккроссингу, який задовольнить потреби користувачів та сприятиме подальшому розвитку проекту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ СИСТЕМИ

3.1 Інформаційна модель веб системи

Інформаційна модель веб системи є важливою складовою розробки даного проекту, адже вона дозволяє зрозуміти логіку роботи сайту та сформулювати вимоги до цього проекту. Це спосіб, яким інформаційні елементи сайту організовані разом з усіма взаємозв'язками, що дає їм можливість взаємодіяти. Ступінь зручності для користувачів під час використання сайту значно залежить від якості цієї взаємодії.

Схема дозволяє краще зрозуміти які сторінки мають основний функціонал та яку очікуваний результат ми маємо тримати при взаємодії з елементами сторінки.

На представленій схемі рис.3.1 ми бачимо, що основними сторінками є: Головна, Книги, Мої пропозиції, Особистий кабінет.

Детальніше про функціонал сторінок:

Головна сторінка – відображається одразу при переадресації на веб систему, мітить загальну описову інформацію про сервіс, відображення кнопок для переходу на інші сторінки та тілі сторінки відображаються рекомендовані книги, які формуються на основі історії замовлень користувача.

Книги – сторінка містить функцію фільтрації та сортування, для зручності вибору та полегшення пошуку книг.

Мої пропозиції – відображаються запити, які надходять він інших користувачів з пропозицією обміну книжками. Запити містять описову частину, ім’я користувача, фото книги та дві кнопки «Підтвердити» та «Відхилити» , при взаємодії з кнопкою «Підтвердити» - відкрається діалогове вікно для можливості обговорення запиту.

Мій кабінет – є доступні функції для зареєстрованого користувача та користувача, який ще не зареєстрований. Не зареєстрований користувач – відображаються дві кнопки «Увійти» та «Зареєструватись». При взаємодії з цими елементами, відкривається відповідна форма для вводу даних

користувача для реєстрації або входу. Зареєстрований користувач має доступ до особистої інформації через особистий кабінет та можливість редагувати дані.

Мої книги – відображаються книги, які користувач вже додав в систему та клікабельна кнопка «Додати книгу». Для додавання книги розроблена форма з обов'язковими полями.

Рисунок 3.1 – Логічна схема веб системи буккроксінгу

Для докладного розроблення сценарію роботи онлайн буккросингу та уточнення функціоналу системи, було розроблено діаграму опису ролей, які будуть використовувати функціонал (табл.3.1). Ця діаграма визначає ключові ролі користувачів та їх можливості в системі:

- Зареєстрований користувач;
- Незареєстрований користувач;
- Адміністратор;
- Система.

Таблиця 3.1 – Опис ролей у використанні онлайн буккросингу

Роль	Опис
Зареєстрований користувач	Користувач, який створив обліковий запис у системі буккросингу. Має можливість авторизації, додавання та видалення книг, перегляду та редагування свого профілю, участі в обмінах книгами.
Не зареєстрований користувач	Користувач, який переглядає доступні книги та може здійснити реєстрацію для отримання додаткових можливостей. Має обмежений функціонал порівняно із зареєстрованим користувачем.
Система	Автоматизований процес управління та взаємодії всіх користувачів. Забезпечує реєстрацію, авторизацію, додавання та видалення книг, проведення обмінів, взаємодію з базою даних та надання інформаційних сервісів.
Адміністратор	Особа, відповідальна за адміністрування системи буккросингу. Має повний доступ до адміністративних функцій, таких як управління користувачами, вирішення конфліктів, підтримка користувачів.

Визначивши ролі користувачів системи, було розроблено сценарій використання.

Таблиця 3.2 - Опис варіантів використання

Роль	Опис доступного функціоналу
Зареєстрований користувач	Авторизується у системі
	Редагує свій профіль та інформацію про книги
	Відправляє запит на обмін книгою з іним користувачем
	Ведення спілкування з іншими користувачеми у внутрішньому чаті
	Переглядає книги в системі
	Додає нову книгу в свій профіль
Не зареєстрований користувач	Переглядає доступні книги без авторизації
	Реєструється в системі
Адміністратор	Управляє користувачами: блокує, розблоковує або видаляє облікові записи
	Надає підтримку користувачам у разі потреби
Система	Реєструє та авторизує користувачів.
	Управляє каталогом книг та їх доступністю.
	Забезпечує безпеку даних та взаємодії в системі.

Кожен сценарій відображає ключові кроки та можливості, які доступні користувачам відповідно до їхнього статусу та ролі в платформі (рис.3.2).

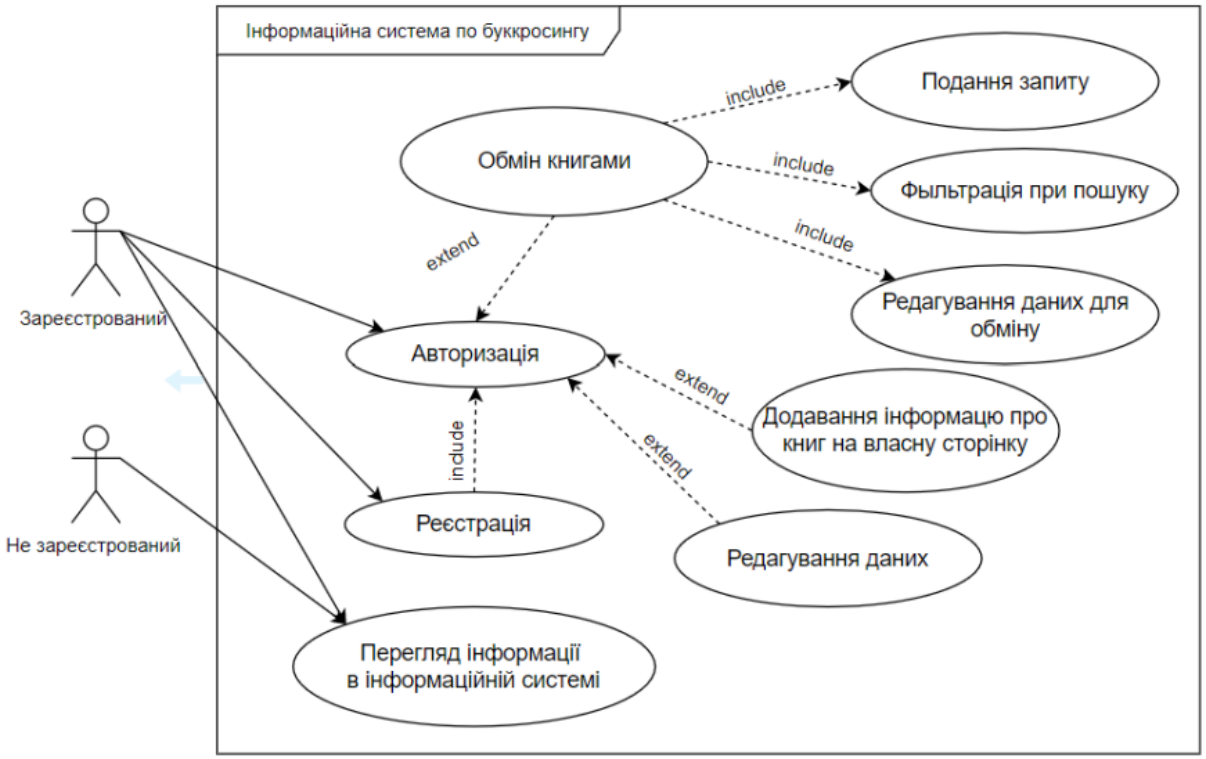

Рисунок 3.2 – Діаграма використання системи онлайн буккросингу

3.2 Створення бази даних

Під час розробки, ми використали систему керування реляційними базами даних MySQL, яка є вільною та з відкритим вихідним кодом. Ця система була розроблена як альтернатива комерційним системам і знайшла широке застосування в створенні динамічних веб-сторінок, завдяки своїй підтримці для різних мов програмування.

Взаємодія між базою даних та веб-сайтом відбувається за допомогою мови програмування PHP. Спочатку відбувається підключення до бази даних, а потім за допомогою мови запитів SQL здійснюються операції видалення чи зміни даних в базі - CRUD (create, read, update, delete). Структура бази даних відображена схемою (рис. 3.2).

Кожна таблиця в базі містить певну категорію інформації, включаючи дані про користувачів, книг, авторів, поточні сесії, матеріали додані до сайту, та створені статичні сторінки.

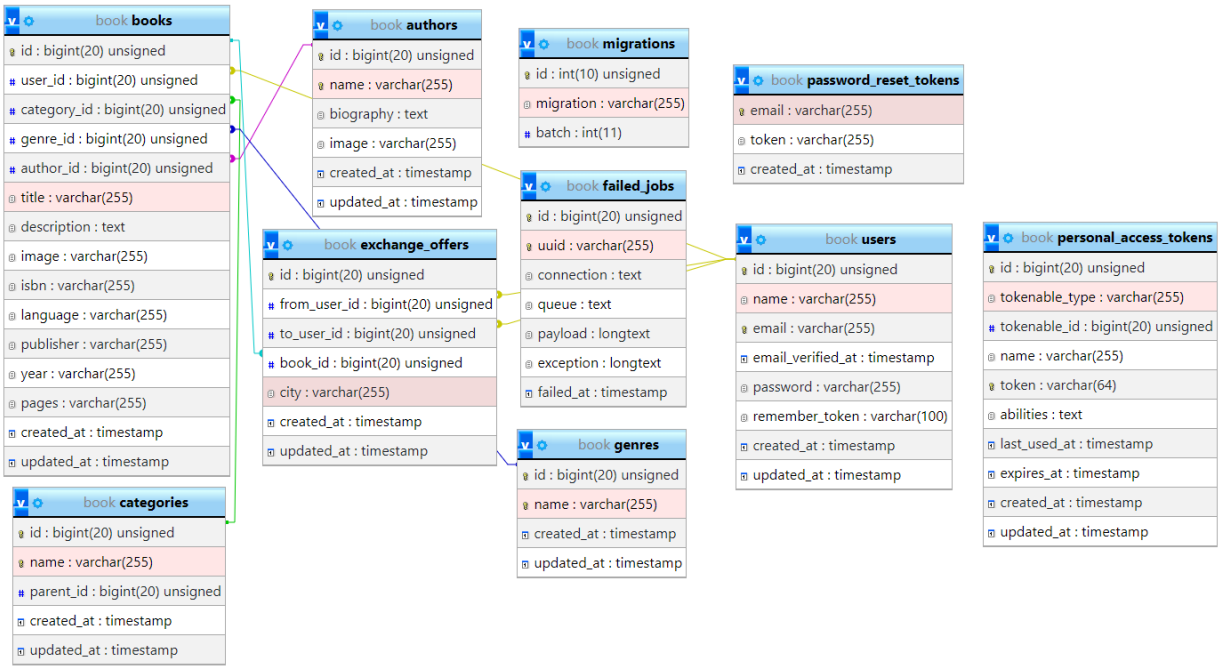

Рисунок 3.2 – Структура бази даних

Таблиця 3.3 – Структура бази данних для буккросингу

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	books	id	Ідентифікатор книги	BIGINT(20)	PK	Не порожній
		user_id	Ідентифікатор користувача	BIGINT(20)	FK	Не порожній
		category_id	Ідентифікатор категорії	BIGINT(20)	FK	Не порожній
		genre_id	Ідентифікатор жанру	BIGINT(20)	FK	Не порожній
		author_id	Ідентифікатор автора	BIGINT(20)	FK	Не порожній
		title	Назва книги	VARCHAR(255)		Не порожній
		description	Опис книги	TEXT		
		image	Зображення	VARCHAR(255)		Не порожній
		isbn	ISBN	VARCHAR(255)		
		language	Мова	VARCHAR(255)		Не порожній
		publisher	Видавець	VARCHAR(255)		
		year	Рік видання	VARCHAR(255)		
		pages	Кількість сторінок	VARCHAR(255)		
		created_at	Дата створення запису	TIMESTAMP		

Продовження таблиці 3.3

		updated_at	Дата оновлення запису	TIMESTAMP		
2	authors	id	Ідентифікатор автора	BIGINT(20)	PK	Не порожній
		name	Ім'я автора	VARCHAR(255)	UNI	Не порожній
		biography	Біографія	TEXT		
		image	Зображення	VARCHAR(255)		
		created_at	Дата створення запису	TIMESTAMP		
		updated_at	Дата оновлення запису	TIMESTAMP		
3	categories	id	Ідентифікатор категорії	BIGINT(20)	PK	Не порожній
		name	Назва категорії	VARCHAR(255)	UNI	Не порожній
		parent_id	Ідентифікатор батьківської кат.	BIGINT(20)	FK	
		created_at	Дата створення запису	TIMESTAMP		
		updated_at	Дата оновлення запису	TIMESTAMP		

Продовження таблиці 3.3

4	exchange_offers	id	Ідентифікатор пропозиції обміну	BIGINT(20)	PK	Не порожній
		from_user_id	Ідентифікатор користувача, який пропонує обмін	BIGINT(20)	FK	Не порожній
		to_user_id	Ідентифікатор користувача, якому пропонують обмін	BIGINT(20)	FK	Не порожній
		book_id	Ідентифікатор книги	BIGINT(20)	FK	Не порожній
		city	Місто	VARCHAR(255)		Не порожній
		created_at	Дата створення запису	TIMESTAMP		
		updated_at	Дата оновлення запису	TIMESTAMP		
5	genres	id	Ідентифікатор жанру	BIGINT(20)	PK	Не порожній
		name	Назва жанру	VARCHAR(255)	UNI	Не порожній
		created_at	Дата створення запису	TIMESTAMP		
		updated_at	Дата оновлення запису	TIMESTAMP		

Продовження таблиці 3.3

6	failed_jobs	id	Ідентифікатор запису	BIGINT(20)	PK	Не порожній
		uuid	Унікальний ідентифікатор	VARCHAR(255)	UNI	Не порожній
		connection	З'єднання	TEXT		Не порожній
		queue	Черга	TEXT		Не порожній
		payload	Дані	LONGTEXT		Не порожній
		exception	Виняток	LONGTEXT		Не порожній
		failed_at	Час виникнення помилки	TIMESTAMP		Не порожній
7	users	id	Ідентифікатор користувача	BIGINT(20)	PK	Не порожній
		name	Ім'я користувача	VARCHAR(255)		Не порожній
		email	Електронна пошта	VARCHAR(255)	UNI	Не порожній
		email_verified_at	Підтвердження електронної пошти	TIMESTAMP		
		password	Пароль користувача	VARCHAR(255)		Не порожній

Продовження таблиці 3.3

		remember_token	Токен для запам'ятовування	VARCHAR(100)		
		created_at	Дата створення запису	TIMESTAMP		
		updated_at	Дата оновлення запису	TIMESTAMP		
8	migrations	id	Ідентифікатор міграції	INT(10)	PK	Не порожній
		migration	Назва міграції	VARCHAR(255)		Не порожній
		Пакет	INT(11)			Не порожній
9	password_reset_tokens	email	Електронна пошта	VARCHAR(255)	PK	Не порожній
		token	Токен для відновлення пароля	VARCHAR(255)		Не порожній
		created_at	Дата створення запису	TIMESTAMP		
10	personal_access_tokens	id	Ідентифікатор токену	BIGINT(20)	PK	Не порожній
		tokenable_type	Тип сутності, якій належить токен	VARCHAR(255)		Не порожній
		tokenable_id	Ідентифікатор сутності	BIGINT(20)	FK	Не порожній
		name	Назва токену	VARCHAR(255)		Не порожній

Продовження таблиці 3.3

		token	Токен	VARCHAR(64)	UNI	Не порожній
		abilities	Можливості	TEXT		
		last_used_at	Останнє використання	TIMESTAMP		
		expires_at	Термін дії	TIMESTAMP		
		created_at	Дата створення запису	TIMESTAMP		
		updated_at	Дата оновлення запису	TIMESTAMP		

3.3 Налаштування бібліотек та модулів

Для розробки інформаційної системи для буккросингу було обрано такі фрейморки, як:

- Laravel для розробки частини back-end;
- Vue.js для розробки частини front-end.

Для початку розробки веб системи, важливим етапом є налаштування бібліотек та модулів. На рисунку 3.3 відображені файли, які використовувались для в процесі розробки.

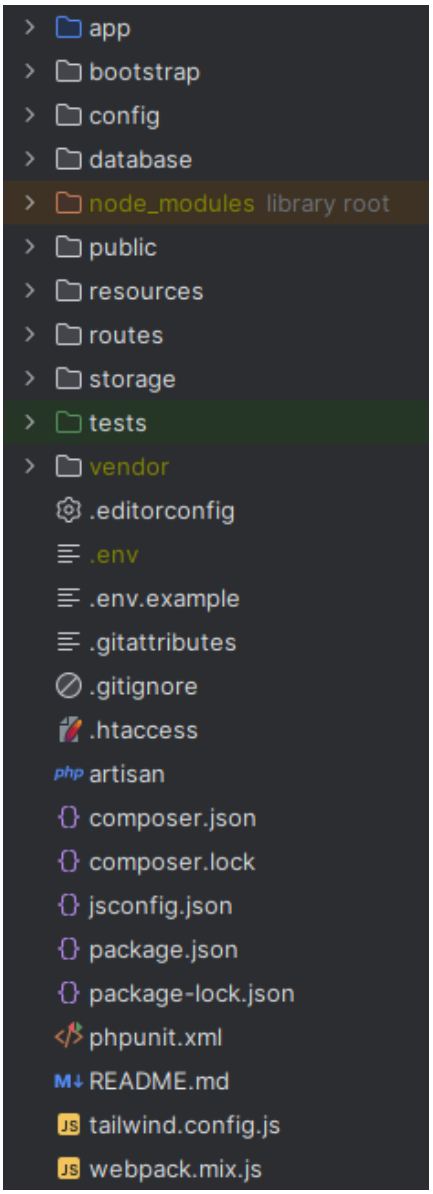

Рисунок 3.3 – Файли, які були створені для проекту

У процесі розробки використовувались файли: app, bootstrap, config, database, lang, public, resources, routes, storage, tests.

Встановлюємо бібліотеки composer.json для backend та package.json для frontend та прописуємо умови інформаційної системи буккроксингу.

3.4 Практичне використання інформаційної системи користувачем

Для початку використання інформаційної системи користувачем необхідно перейти за посиланням http://book.codexgen.com/ . На рисунку 3.4 відобажається головна сторінка, на яку потрапляє користувач.

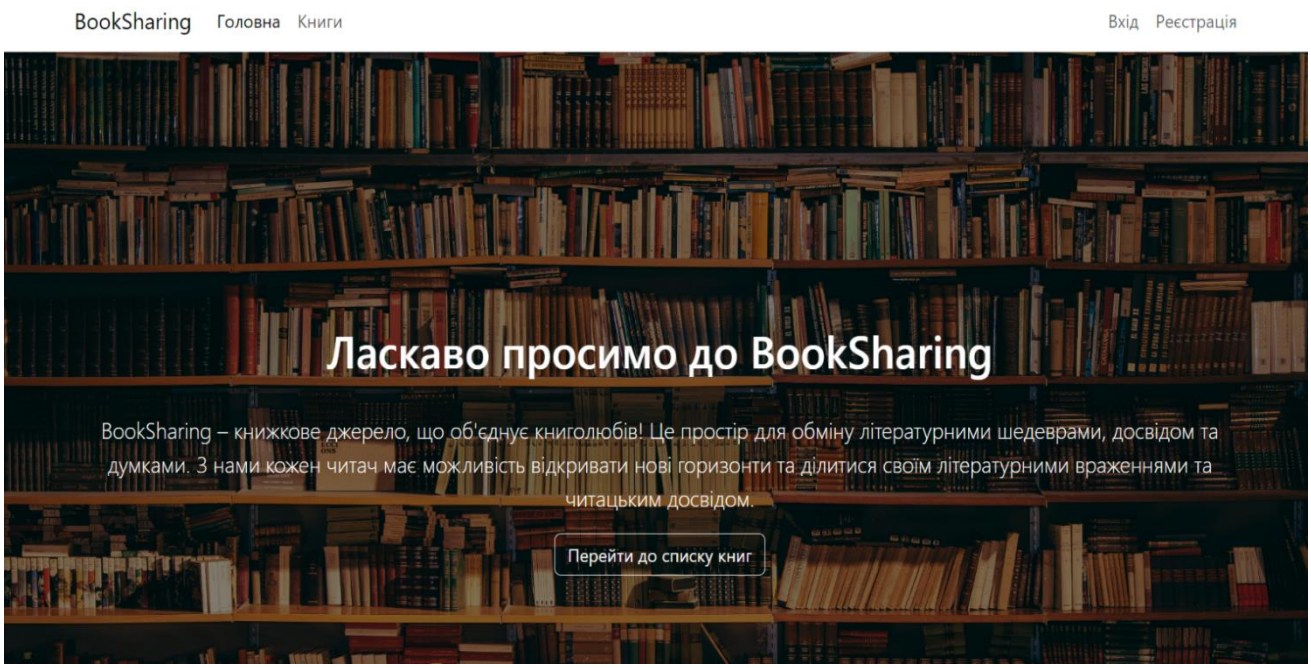

Рисунок 3.4. – Головна сторінка інформаційної системи BookSharing

На презентованій сторінці виконаний мінімалістичний стиль, що дозволяє користувачеві зосередитись на основній інформації про систему буккросингу та ознайомитись з функціоналом, не відволікаючись на попереднє введення даних.

Не зареєстрований користувач може ознайомитись з системою, переглянути примірники книг, які доступні на сайті та зареєстуватись.

Для реєстрації необхідно натиснути на клікабельну кнопку в хедері сторінки «Реєстрація». Для успішно виконаної дії користувач має заповнити відповідні поля з даними (рисунок 3.5).

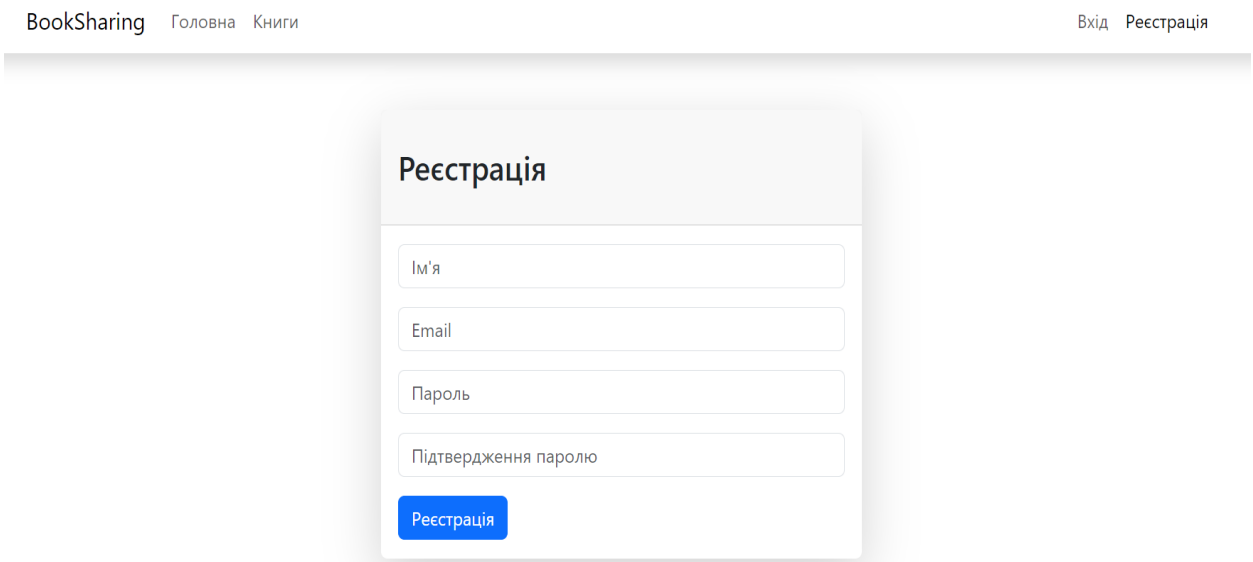

Рисунок 3.5. – Форма реєстрації системи буккроксингу

Всі поля є обов'язковими для заповнення. В разі виявлення системою невідповідності – повідомляє користувачеві, що саме необхідно виконати для завершення успішної реєстрації (рисунок 3.6).

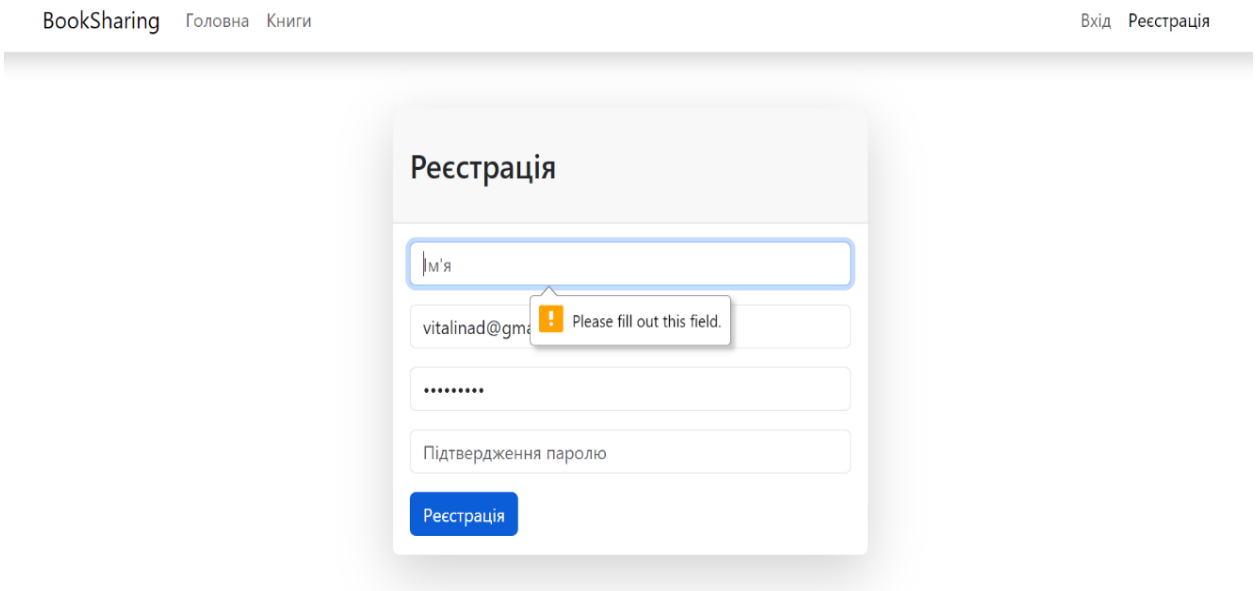

Рисунок 3.6. – Валідація введених даних системою

На рисунку 3.7 відображається форма для авторизації після успішної реєстрації користувачем.

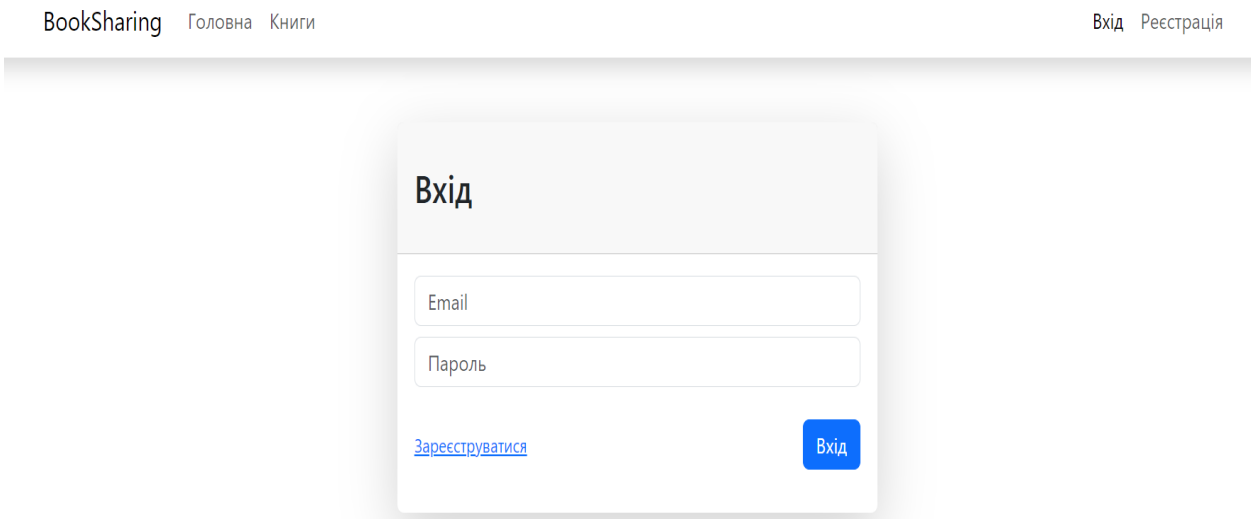

Рисунок 3.7. – Форма для авторизації зареєстрованого користувача

Зареєстрований користувач переадресовується на головну сторінку інформаційної системи для буккроксингу (рисунок 3.8).

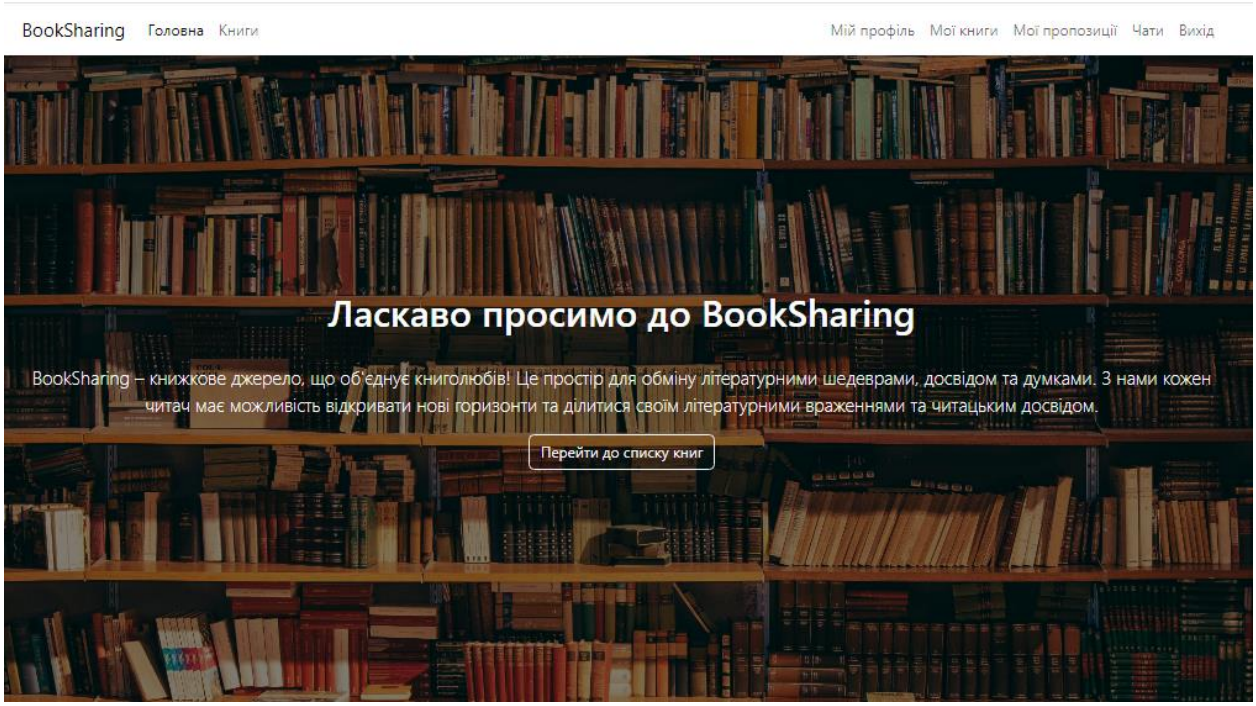

Рисунок 3.8. – Головна сторінка авторизованого користувача системи

Зареєстрованому користувачу необхідно завершити налаштування особистого кабінету (рис. 3.9).

Мій кабінет містить поля, доступні для редагування:

- Зміна паролю (поточний пароль, новий, підтвердження нового паролю).
- Поле, де користувач може додати інформацію про свої літературні вподобання, читацький досвід, улюблених авторім, тощо.
- Аватар, доступна функція завантаження файлів типу PNG та JFIF.

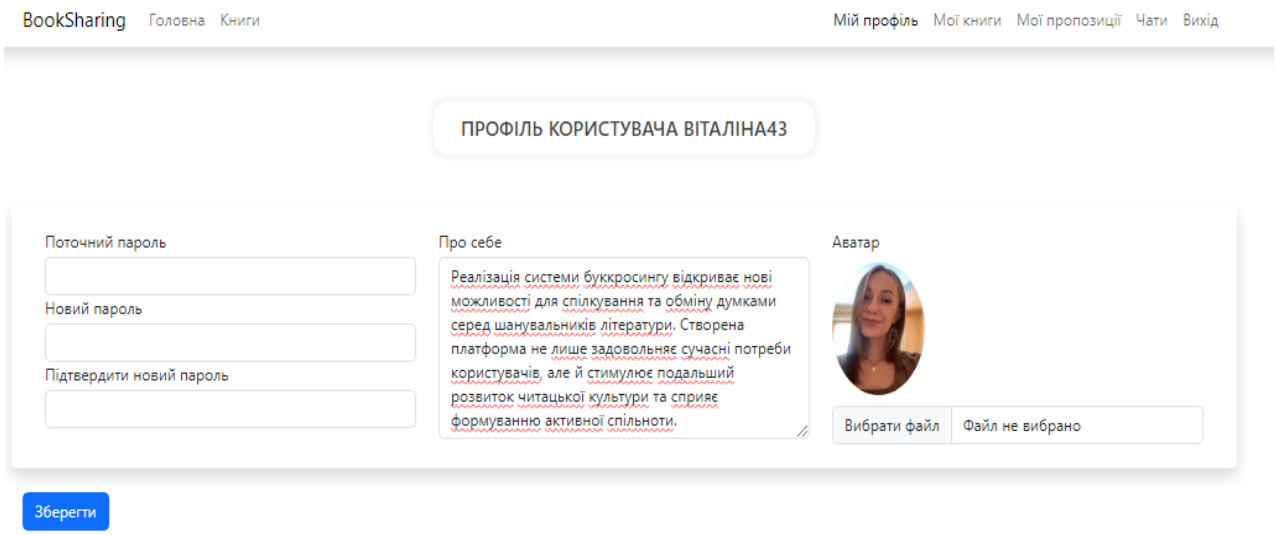

Рисунок 3.9 – Сторінка «Мій профіль»

Для використання функціїї «Запропонувати обмін» необхідно обрати книгу для рекомендаціїї іншому користувачеві. На рисунку 3.10 відображена форма для додавання нової книги в особистий кабінет «Мої книги».

Для довання книги необхідно натиснути та вести дані про книгу в обов’язкові поля:

- Автор;
- Рік видання;
- Фото;
- Жанр;
- Категорія;
- Мова;

- Видавництво;
- Короткий опис.

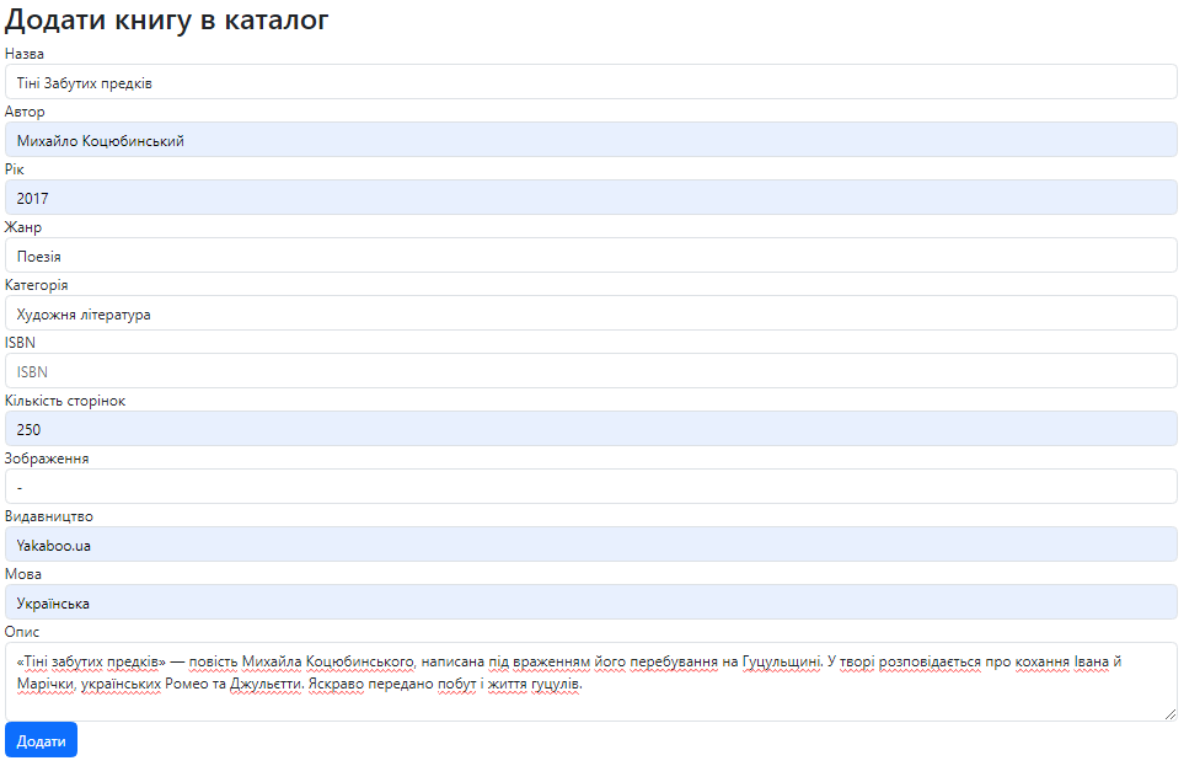

Рисунок 3.10 – Форма додавання книги на сторнці «Мої книги»

Проте для зручності можна ще скористатись функцією «додати примірник», у відповідне поле ввести назву книги або автора та перевірити наявні примірники в системі. Якщо такі книги вже є в базі даних, то система надає можливість додати примірник в особистий кабінет.

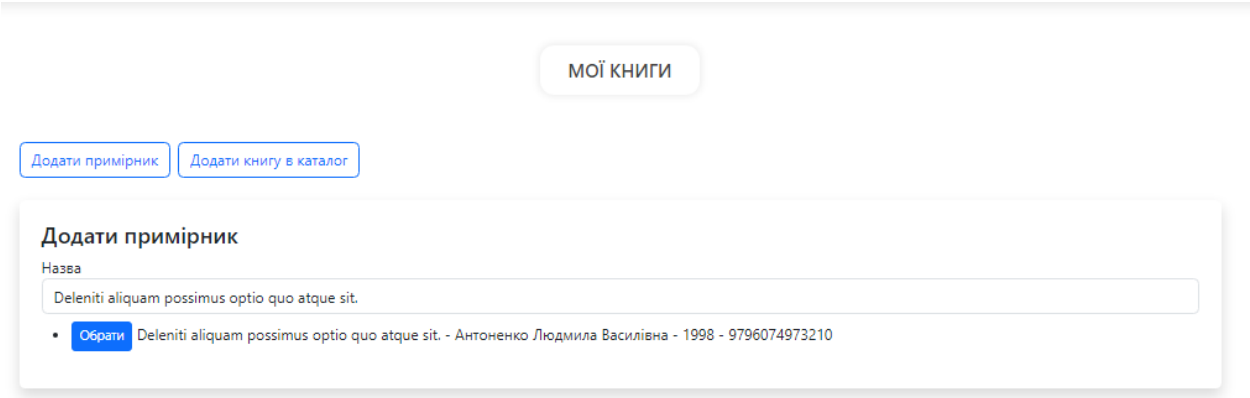

Рисунок 3.11 – Додавання книги за допомогою функції «Додати примірник»

Додані книги відображаються на «Мої книги», де користувач може переглядати книги, які вже є в особистому кабінеті.

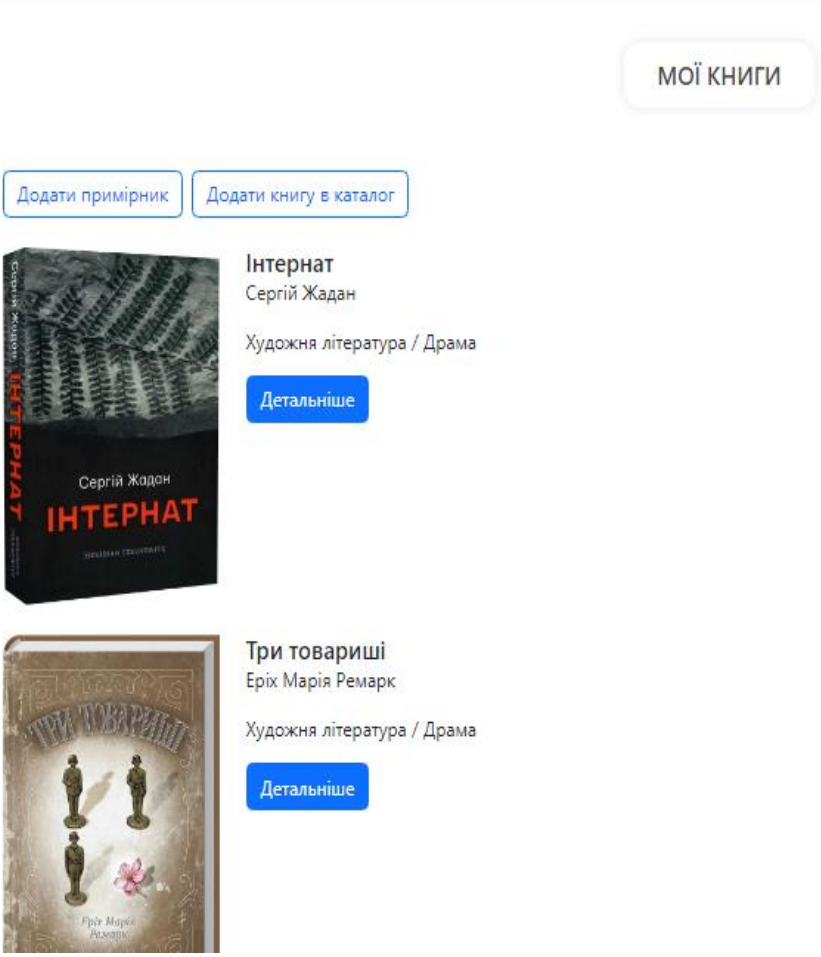

Рисунок 3.12 – Сторінка «Мої книги»

Для переходу в загальний каталог, необхідно натиснути кнопку «Книги», яка знаходиться в хедері сторінки.

Сторінка «Книги», де користувач може ознайомитись з примірниками які знаходяться в загальному каталозі (рис.3.13).

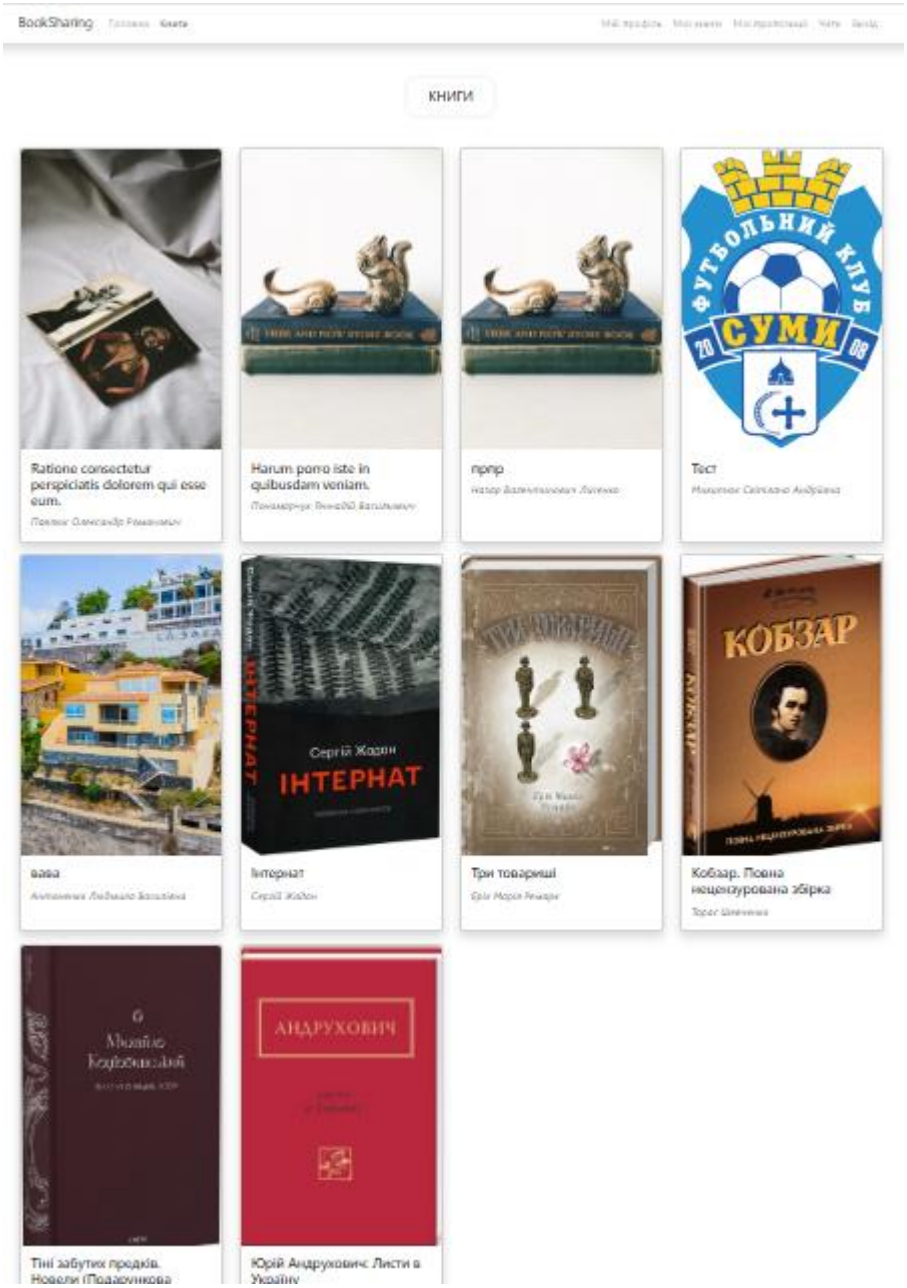

Рисунок 3.13 – Загальний каталог, сторінка «Книги»

Для ознайомлення з більш детальною інформацією про книгу, необхідно натиснути на неї, після чого відбувається переадресація на сторінку з детальним описом книги.

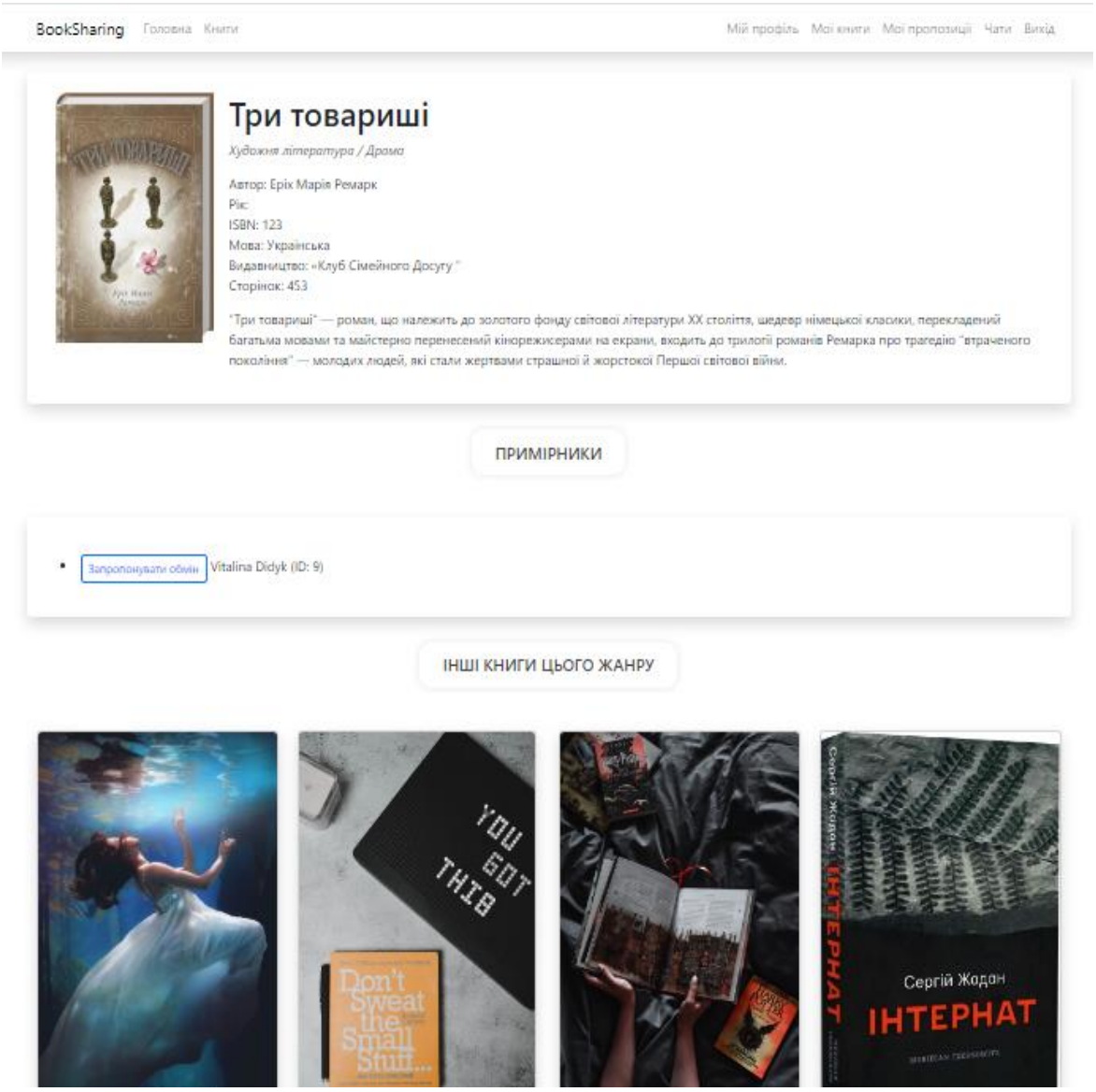

Рисунок 3.14 – Відображення інформації про книгу

На презентованій сторінці відображається наступна інформація про книгу:

- Автор;
- Рік видання;
- Видавництво;
- Мова;
- Кількість сторінок.

Для обміну книгою необхідно скористатись клікабельною кнопкою «Запропонувати обін». Після чого відображається відправлений запит та можливість відкрити вікно чату для ведення діалогу з власником примірника.

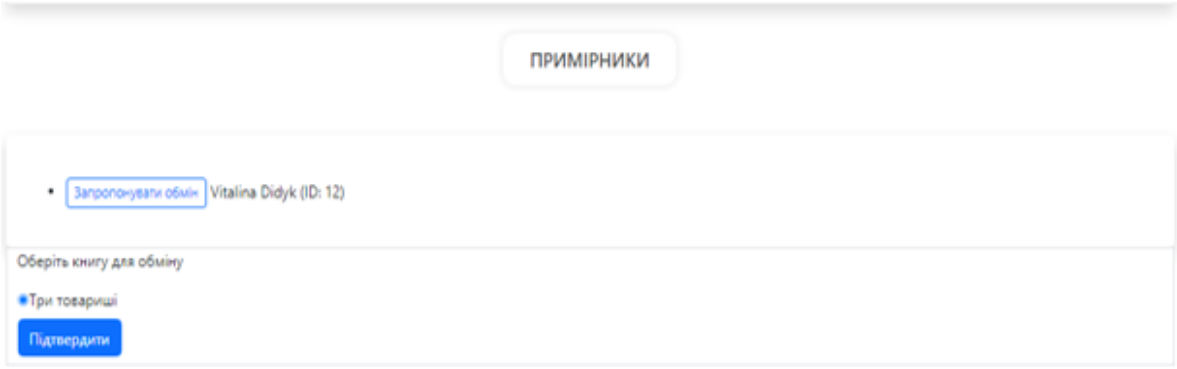

Рисунок 3.15 – Відправлення запиту на обмін книгою

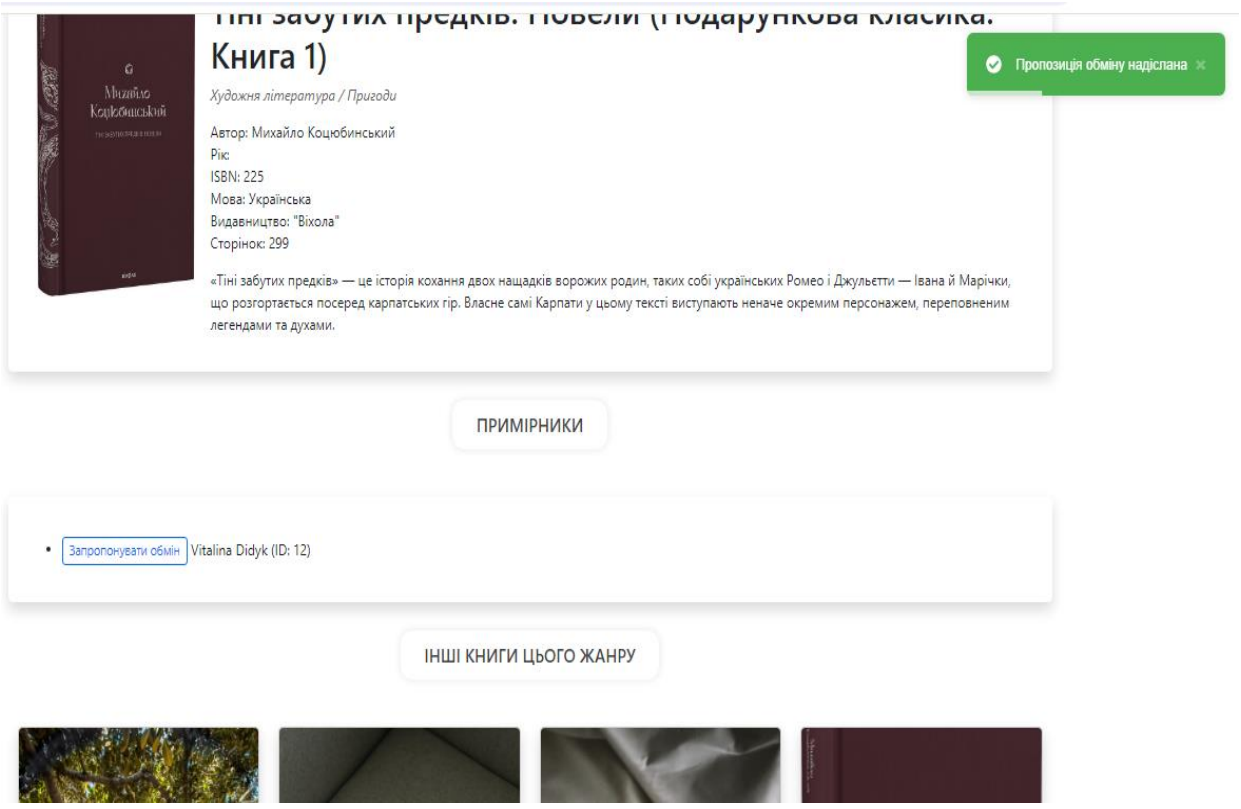

Рисунок 3.16 – Підтвердження запиту на обмін книгою

Для зручності, для випадку коли примірника не має в наявності або мова книги не підходить користувачеві - на представленій сторінці відображається поле «Інші книжки цього жанру». Користувач зможе переглянути інформацію про наявні книги цього жанру та запропонувати обмін.

Після відправлення запиту на обмін книгою, на сторінці «Мої пропозиції» з’являється інформація про запит від іншого користувача та є можливість прийняти або відхилити запит з обміну (рис.3.17).

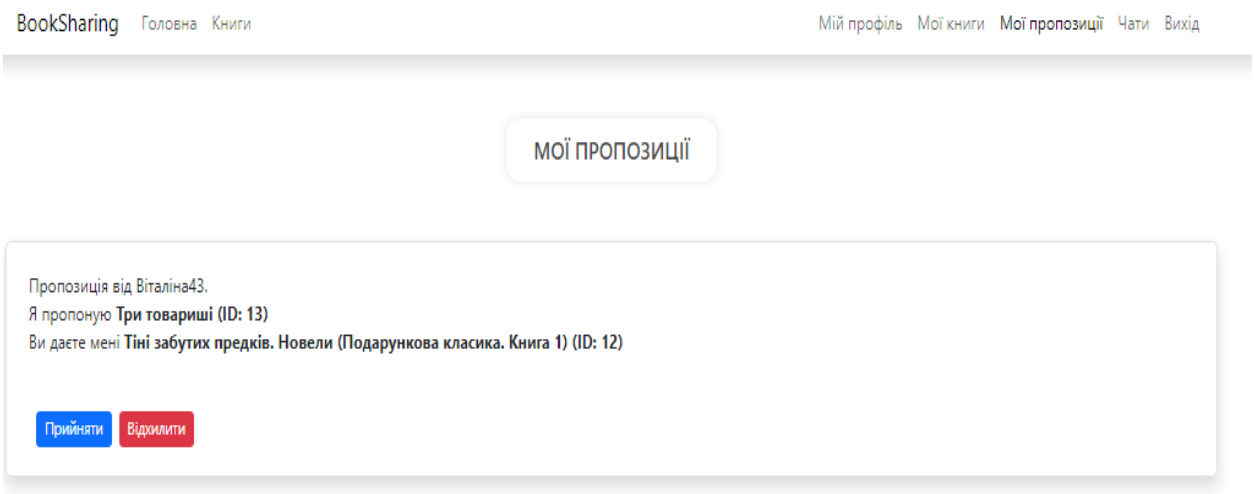

Рисунок 3.17 – Сторінка «Мої пропозиції»

Прийнявши запит від користувача, відкривається можливість спілкування з іншим користувачем системи, через внутрішній чат. Для цього, необхідно перейти на сторінку «Чат», де вже автоматично підтягнеться користувач, який запропонував обмін (рис.3.18). Таким чином, користувачі можуть домовитись, про відправлення книги та в подальшому використовувати чат для обміну враженнями.

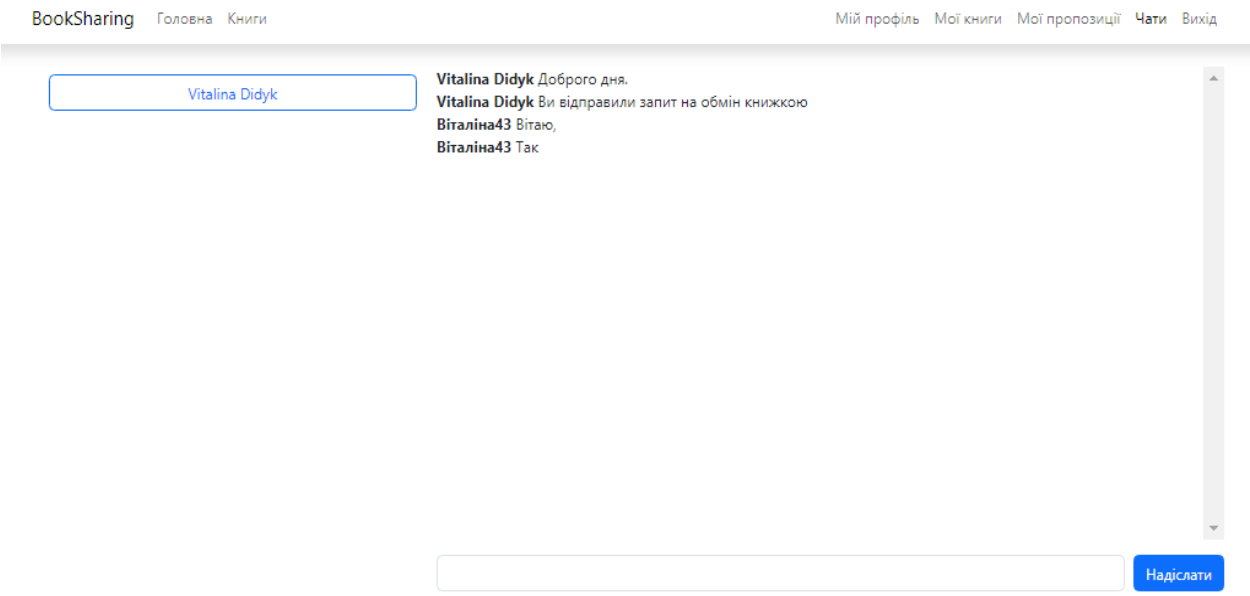

Рисунок 3.18 – Внутрішній чат, для спілкування користувачами

Після успішних операцій обміну, система буде автоматично на головній сторінці рекомендувати користувачеві книги, відповідно до жанрів, які були обрані.

ВИСНОВКИ

У результаті виконання кваліфікаційної магістерської роботи розроблено інформаційну систему онлайн буккросингу та виконано усі поставлені завдання.

1) Проведено огляд та аналіз сучасної джерел інформації в контексті розробки інформаційних систем, з метою сформулювання завдань для подальшої реалізації.
2) Досліджено переваги та недоліки існуючих систем, визначено ключові аспекти, які враховововано при розробці власної технології.
3) Сформовано функціональні та не функціональні вимоги до проєкту.
4) Проаналізовано інструменти та обрано інструменти (Laravel для бекенду та Vue.js для фронтенду). Ця комбінація дозволила успішно створити оптимізовану платформу для задоволення потреб користувачів.
5) Спроектовано логічну схему та діаграму використання інформаційної технології.
6) Розроблено та налаштовано базу даних для системи.
7) Програмно реалізовано систему буккросингу, яка включає в себе ключові функції, спрямовані на створення зручного та ефективного середовища для обміну та взаємодії серед читачів.

Відтворено ключові функції, спрямовані на оптимізацію та полегшення обміну книг користувачами. Розроблена інформаційна система має свої переваги, проте існують напрямки для подальших вдосконалень. Серед них варто виділити поліпшення інтерфейсу, оптимізацію масштабування та локалізації, а також налаштування системи сповіщень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Секрети буккросингу [Електронний ресурс] – URL: https://zaxid.net/sekreti_bukkrosingu_lviv_n1092127 (дата звернення: 06.10.2023);
2. Сайт аналог «Bookcrossing-ua.at.ua» [Електронний ресурс] – URL: https://bookcrossing-ua.at.ua (дата звернення: 06.10.2023);
3. Сайт аналог «Goodreads» [Електронний ресурс] – URL: https://www.goodreads.com/ (дата звернення: 06.10.2023);
4. Сайт аналог «Booksaround» [Електронний ресурс] – URL: https://www.booksaround.net/ (дата звернення: 06.10.2023);
5. Телеграм група «Bookcrossing.ua» [Електронний ресурс] – URL: https://t.me/bookcrossing_ua (дата звернення: 06.10.2023);
6. Фреймворки у веб-розробці [Електронний ресурс] - URL: https://highload.today/uk/frejmvorki-u-veb-rozrobtsi-shho-tse-yaki-isnuyut-i-dlya-chogo-potribni/ (дата звернення: 13.10.2023);
7. Scott Rogers: Level Ups! The Guide to Great Video Game Design / Wiley, 2020. 560 с.
8. Kathy Sierra, Trisha Gee, Bert Bates: Head First Java: A Brain-Friendly Guide / O'Reilly Media, 2022. 752 с.
9. Двірничук К.В., Вацек Д.О.: Навчальний посібник веб-програмування та веб-дизайну, 2022. 13с.
10. РОЗРОБКА ВЕБ-ДОДАТКІВ ЗА ДОПОМОГОЮ ФРЕЙМВОРКА CODEIGNITER [Електронний ресурс] – URL: https://conf.ztu.edu.ua/wp-content/uploads/2017/06/163-2.pdf (дата звернення: 27.10.2023);
11. Kohana: Swift PHP Framework [Електронний ресурс] – URL: https://coderlessons.com/articles/veb-razrabotka-articles/kohana-swift-php-framework (дата звернення: 27.10.2023);

12. [Електронний ресурс] – URL: https://wezom.com.ua/ua/blog/laravel-vs-yii-framework-chto-luchshe-dlja-vashego-biznesa (дата звернення: 28.10.2023);
13. The PHP Framework for Web Artisans [Електронний ресурс] - URL: https://laravel.com/ (дата звернення: 30.10.2023);
14. 17 переваг фреймворку Laravel [Електронний ресурс] - URL: https://icstudio.online/post/17-perevag-frejmvorku-laravel (дата звернення: 30.10.2023);
15. Dronov V. Rapid development on PHP 2021. – 394 с.
16. ПРО ФРЕЙМВОРК VUE.JS [Електронний ресурс] - URL: https://conf.ztu.edu.ua/wp-content/uploads/2017/11/21.pdf (дата звернення: 30.10.2023);
17. Merih Taze: Engineers Survival Guide: Advice, tactics, and tricks After a decade of working at Facebook, Snapchat, and Microsoft Paperback / Merih Taze, 2021. – 245 с
18. Прототип або прототипування сайту [Електронний ресурс] - URL: https://www.markweb.pro/prototip-abo-prototipuvannya-sajtu-prototip-sajtu-priklad/ (дата звернення: 01.11.2023);
19. Веб-дизайн. Інформаційна структура сайту [Електронний ресурс] - URL: https://webstudio2u.net/ua/design-web/443-information-structure.html (дата звернення: 01.11.2023);
20. Структура сайту: що це таке, види та приклади - як створити структуру для інтернет-магазину [Електронний ресурс]. URL: https://elit-web.ua/ua/blog/kak-vyvesti-tekuschuju-strukturu-sajta (дата звернення: 20.11.2023).
21. Bringing MySQL to the web [Електронний ресурс] - URL: https://www.phpmyadmin.net/ (дата звернення: 05.11.2023);
22. "MySQL Explained: Your Step-by-Step Guide" by Andrew Comeau 2017. – 211с.

23. Martin Fowler: Refactoring: Improving the Design of Existing Code / AddisonWesley Professional, 2018. – 448 с.
24. Scott Rogers: Level Ups! The Guide to Great Video Game Design / Wiley, 2020. – 560 с.
25. Nicole Forsgren PhD, Jez Humble, Gene Kim: Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations / IT Revolution Press, 2018. – 289 с.
26. Martin the Robert: Clean Code: A Handbook of Agile Software Craftsmanship / Pearson, 2017. – 1214 с.
27. useForm | React Hook Form - Simple React forms validation [Електронний ресурс]. URL: https://www.react-hook-form.com/api/useform/ (дата звернення: 05.011.2023).
28. JSON Web Tokens - jwt.io [Електронний ресурс]. URL: https://jwt.io/ (дата звернення: 10.11.2023).
29. JavaScript [Електронний ресурс] - URL: https://uk.javascript.info/ (дата звернення: 28.10.2023);

ДОДАТОК А
КОД РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

BookController.php

Контролер книг

```
<?php

namespace App\Http\Controllers;

use App\Models\Author;
use App\Models\Book;
use App\Models\BookInstance;
use App\Models\Category;
use App\Models\Genre;
use Illuminate\Http\Request;

class BookController extends Controller
{
   public function index() {
      //Get books with category and genre and paginate them
      $books = Book::with('category', 'genre', 'author')->paginate(12);
      return response()->json($books);
     }

     public function last() {
        $books = Book::with('category', 'genre', 'author')->latest()->take(8)->get();
        return response()->json($books);
     }

     public function byGenre($id) {
        $books = Book::with('category', 'genre', 'author')->where('genre_id', $id)->take(4)->get();
        return response()->json($books);
     }

     public function my() {
        $user = auth()->user();
        $books = BookInstance::with('book.category', 'book.genre', 'book.author')-
  >where('current_user_id', $user->id)->get();
```

```
    return response()->json($books);
}

public function show($id) {
    $book = Book::with('category', 'genre', 'author', 'instance', 'instance.current')->findOrFail($id);
    return response()->json($book);
}

public function createInstance(Request $request) {
    $request->validate([
        'book_id' => 'required|exists:books,id',
    ]);

    $book = Book::findOrFail($request->input('book_id'));

    $instance = BookInstance::create([
        'book_id' => $book->id,
        'user_owner_id' => auth()->user()->id,
        'current_user_id' => auth()->user()->id,
        'description' => $request->input('description') ?? '',
    ]);

    return response()->json($instance, 201);
}

public function store(Request $request) {
    $request->validate([
        'title' => 'required|string|max:255',
        'author_id' => 'required|exists:authors,id',
        'category_id' => 'required|exists:categories,id',
        'genre_id' => 'required|exists:genres,id',
        'description' => 'string',
```

```
        'image' => 'image|mimes:jpeg,png,jpg,gif,svg|max:10240',
    ]);

    $data = $request->all();

    if ($request->hasFile('image')) {
        $path = $request->file('image')->store('books', 'public');
        //full url to image
        $data['image'] = '/storage/' . $path;
    }

    $data['user_id'] = auth()->user()->id;

    $book = Book::create($data);

    return response()->json($book, 201);
}

public function search(Request $request) {
    $request->validate([
        'term' => 'required|string|max:255',
    ]);

    $query = $request->input('term');

    $books = Book::with('author')
        ->where('title', 'LIKE', "%{$query}%")
        ->orWhere('isbn', 'LIKE', "%{$query}%")
        ->take(10)
        ->get();

    return response()->json($books);
}

public function genres() {
    $genres = Genre::all();
    return response()->json($genres);
}
```

```
    public function authors(Request $request) {
        $authors = Author::where('name', 'LIKE', "%{$request->input('name')}%")->take(15)->get();
        return response()->json($authors);
    }

    public function categories() {
        $categories = Category::all();
        return response()->json($categories);
    }

    // Add methods for creating, updating, and deleting books as per your needs
}
```

ChatController.php

Контролер чату

```
<?php

// app/Http/Controllers/ChatController.php

namespace App\Http\Controllers;

use App\Events\MessageSent;
use App\Models\ExchangeOffer;
use App\Models\Message;
use App\Models\User;
use Illuminate\Http\Request;

class ChatController extends Controller {

    public function getUsers() {
        $userId = auth()->id();

$exchangeUserIds = ExchangeOffer::where('from_user_id', $userId)
    ->orWhere('to_user_id', $userId)
    ->get()
```

```
        ->get()
        ->map(function ($offer) use ($userId) {
            return $offer->from_user_id == $userId ? $offer->to_user_id : $offer->from_user_id;
        })
        ->toArray();

    $uniqueUserIds = array_unique(array_merge($exchangeUserIds, $messageUserIds));
    $uniqueUserIds = array_diff($uniqueUserIds, [$userId]);

    $users = User::whereIn('id', $uniqueUserIds)->get();

    return response()->json($users, 200);
}

public function fetchMessages(Request $request) {
    $messages = Message::where('user_id', $request->userId)
        ->orWhere('user_id', $request->toUserId)
        ->orWhere('to_user_id', $request->userId)
        ->orWhere('to_user_id', $request->toUserId)
        ->orderBy('id', 'ASC')->with('user')->get();

    return response()->json($messages, 200);

     public function sendMessage(Request $request)
    {
        $user = auth()->user();
        $message = $user->messages()->create([
            'message' => $request->message,
            'to_user_id' => $request->toUserId
        ]);

        // Ensure you're loading the user relationship if it's not automatically loaded
        $message->load('user');

        broadcast(new MessageSent($message, $request->toUserId))->toOthers();

          // Return the message and the user in separate keys
          return ['message' => $message, 'user' => $user];
      }

  }
```

ExchangeOfferController.php

Контролер обміну книгами

```
<?php

namespace App\Http\Controllers;

use App\Models\BookInstance;
use App\Models\ExchangeOffer;
use Illuminate\Http\Request;

class ExchangeOfferController extends Controller {

    public function index() {
        $offers = ExchangeOffer::all();

        return response()->json($offers, 200);
    }

    public function my() {
        $user = auth()->user();
        $offers = ExchangeOffer::with('give_book.book', 'take_book.book', 'fromUser', 'toUser')-
>where('to_user_id', $user->id)->get();

        return response()->json($offers, 200);
    }

    public function store(Request $request) {
        $request->validate([
            'give_book_instance_id' => 'required|exists:book_instances,id',
            'take_book_instance_id' => 'required|exists:book_instances,id',
            'city' => 'string|max:255',
        ]);
    $give_book_instance = BookInstance::findOrFail($request->input('give_book_instance_id'));
    $take_book_instance = BookInstance::findOrFail($request->input('take_book_instance_id'));
    $data = [
        'give_book_instance_id' => $give_book_instance->id,
        'from_user_id' => auth()->user()->id,
```

```
        'take_book_instance_id' => $take_book_instance->id,
        'to_user_id' => $take_book_instance->current_user_id,
        'city' => 'NONE',
        'status' => 'pending',
    ];

    $offer = ExchangeOffer::create($data);

    return response()->json($offer, 201);
}

// Add additional methods as necessary, such as for viewing and accepting offers

public function accept(Request $request, $id) {
    $offer = ExchangeOffer::findOrFail($id);
        $offer->status = 'accepted';
        $offer->save();
        BookInstance::where('id', $offer->give_book_instance_id)->update(['current_user_id' => $offer->to_user_id]);
        BookInstance::where('id', $offer->take_book_instance_id)->update(['current_user_id' => $offer->from_user_id]);
    }
    public function reject(Request $request, $id) {
        $offer = ExchangeOffer::findOrFail($id);
        $offer->status = 'rejected';
        $offer->save();
    }
}
```

ProfileController.php

Контролер профілю

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
```

```
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
use Inertia\Inertia;
use Inertia\Response;

class ProfileController extends Controller
{
    /**
     * Display the user's profile form.
     */
    public function edit(Request $request): Response
    {
        return Inertia::render('Profile/Edit', [
            'mustVerifyEmail' => $request->user() instanceof MustVerifyEmail,
    'status' => session('status'),
]);
}

/**
 * Update the user's profile information.
 */
public function update(ProfileUpdateRequest $request): RedirectResponse
{
   $request->user()->fill($request->validated());

 if ($request->user()->isDirty('email')) {
    $request->user()->email_verified_at = null;
 }

    $request->user()->save();
```

```
        return Redirect::route('profile.edit');
    }

    /**
     * Delete the user's account.
     */
    public function destroy(Request $request): RedirectResponse
    {
        $request->validate([
            'password' => ['required', 'current_password'],
        ]);

        $user = $request->user();

        Auth::logout();

        $user->delete();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return Redirect::to('/');
    }
}
```

UserController.php

Контролер користувача

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;
```

```
class UserController extends Controller
{
    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\JsonResponse
     */
    public function update(Request $request)
    {
        $user = Auth::user();

        $validatedData = $request->validate([
            'bio' => 'string|max:255',
            'avatar' => 'image|mimes:jpeg,png,jpg,gif,svg|max:10240',
            'current_password' => ['required_with:password', function ($attribute, $value, $fail) use ($user) {
                if ($value && !Hash::check($value, $user->password)) {
                    $fail('Current password is incorrect.');
      }
   }],
   'password' => 'nullable|min:8|confirmed',
]);
  if ($request->hasFile('avatar')) {
     $path = $request->file('avatar')->store('avatars', 'public');
     $validatedData['avatar'] = $path;
  }

  if (isset($validatedData['password'])) {
     $validatedData['password'] = Hash::make($validatedData['password']);
  }

  if (isset($validatedData['bio'])) {
     $validatedData['bio'] = $request->input('bio');
  }
```

```
        $user->update($validatedData);

        return response()->json($user);
    }
}
```

Book.php

Модель книги

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Book extends Model
{
    use HasFactory;

    protected $fillable = [
        'title',
        'description',
        'category_id',
        'genre_id',
        'author_id',
        'image',
        'year',
        'pages',
        'language',
        'publisher',
        'isbn',
        'user_id'
    ];
```

```
    public function category() {
        return $this->belongsTo(Category::class);
    }

    public function genre() {
        return $this->belongsTo(Genre::class);

    }

    public function exchangeOffers() {
        return $this->hasMany(ExchangeOffer::class);
    }

    public function author() {
        return $this->belongsTo(Author::class);
    }

    public function instance() {
        return $this->hasMany(BookInstance::class);
    }
}
```

User.php

Модель користувача

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
```

```
/**
 * The attributes that are mass assignable.
 *
 * @var array<int, string>
 */
protected $fillable = [
    'name',
    'email',
    'password',
    'bio',
    'avatar'
];

/**
 * The attributes that should be hidden for serialization.
 *
 * @var array<int, string>
 */
protected $hidden = [
    'password',
    'remember_token',
];

/**
 * The attributes that should be cast.
 *
 * @var array<string, string>
 */
protected $casts = [
    'email_verified_at' => 'datetime',
    'password' => 'hashed',
];

/**
 * Get the user's avatar full path.
 *
```

```
    * @param  string  $value
    * @return string
    */
   public function getAvatarAttribute($value) {
      return $value ? asset('storage/' . $value) : asset('images/default-avatar.png');
   }

   public function messages() {
      return $this->hasMany(Message::class);
   }
}
```

auth.php

Файл маршрутизації

```
<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;

Route::middleware('guest')->group(function () {
   Route::get('register', [RegisteredUserController::class, 'create'])
         ->name('register');

   Route::post('register', [RegisteredUserController::class, 'store']);

   Route::get('login', [AuthenticatedSessionController::class, 'create'])
         ->name('login');

   Route::post('login', [AuthenticatedSessionController::class, 'store']);
```

```
Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
        ->name('password.request');

Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
        ->name('password.email');

Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
        ->name('password.reset');

  Route::post('reset-password', [NewPasswordController::class, 'store'])
          ->name('password.store');
});

Route::middleware('auth')->group(function () {
  Route::get('verify-email', EmailVerificationPromptController::class)
          ->name('verification.notice');

  Route::get('verify-email/{id}/{hash}', VerifyEmailController::class)
          ->middleware(['signed', 'throttle:6,1'])
          ->name('verification.verify');

  Route::post('email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
          ->middleware('throttle:6,1')
          ->name('verification.send');

  Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
          ->name('password.confirm');

  Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);

  Route::put('password', [PasswordController::class, 'update'])->name('password.update');

  Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
          ->name('logout');
});
```

api.php

Файл маршрутизації всіх шляхів, які використовуються в системі

```
<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\BookController;
use App\Http\Controllers\ChatController;
use App\Http\Controllers\ExchangeOfferController;
use App\Http\Controllers\UserController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Broadcast;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/
Route::middleware('auth:sanctum')->group(function () {
    Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])->name('logout');

    Route::get('books/my', [BookController::class, 'my']);
    Route::post('books/instance', [BookController::class, 'createInstance']);

    // Exchange offers endpoints
    Route::get('/exchange-offers/my', [ExchangeOfferController::class, 'my']);
    Route::post('/exchange-offers', [ExchangeOfferController::class, 'store']);
     Route::post('/exchange-offers/{id}/accept', [ExchangeOfferController::class, 'accept']);
     Route::post('/exchange-offers/{id}/reject', [ExchangeOfferController::class, 'reject']);
     Route::post('/user/update', [UserController::class, 'update']);
```

```
    Route::get('/messages', [ChatController::class, 'fetchMessages']);
    Route::post('/messages', [ChatController::class, 'sendMessage']);
    Route::get('/messages/users', [ChatController::class, 'getUsers']);

});

Route::post('login', [AuthenticatedSessionController::class, 'store']);
Route::post('/register', [RegisteredUserController::class, 'store'])->name('register');

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

// Books endpoints
Route::get('books/last', [BookController::class, 'last']);
Route::get('books/search', [BookController::class, 'search']);
Route::get('books/genre/{id}', [BookController::class, 'byGenre']);
 Route::apiResource('books', BookController::class);

 Route::get('genres', [BookController::class, 'genres']);
 Route::get('authors', [BookController::class, 'authors']);
 Route::get('categories', [BookController::class, 'categories']);

 Route::get('check-auth', function () {
     return response()->json(['message' => 'Authenticated'], 200);
 })->middleware('auth:sanctum');

 Broadcast::routes(['middleware' => ['auth:sanctum']]);
```

router.js

Файл маршрутизації vue.js.

```
 import { createRouter, createWebHistory } from 'vue-router';
 import BookList from '../Components/Pages/Books/BookList.vue';
 import BookDetail from '../Components/Pages/Books/BookDetail.vue';
```

```
import Home from '../Components/Pages/Home.vue';
import Login from '../Components/Pages/Auth/Login.vue';
import Register from '../Components/Pages/Auth/Register.vue';
import ProfileEdit from "../Components/Pages/Auth/Profile/Edit.vue";
import MyBooks from "../Components/Pages/Auth/Profile/MyBooks.vue";
import MyOffers from "../Components/Pages/Auth/Profile/MyOffers.vue";
import Chats from "../Components/Pages/Auth/Profile/Chats.vue";

const routes = [
    {
        path: '/',
        name: 'Home',
        component: Home
    },
    {
   path: '/books',
   name: 'BookList',
   component: BookList
},
{
   path: '/books/:id',
   name: 'BookDetail',
   component: BookDetail,
   props: true
},
{
   path: '/login',
   name: 'Login',

     component: Login,
     meta: {
        guest: true
     }
  },
  {
```

```
    path: '/my-books',
    name: 'MyBooks',
    component: MyBooks,
    meta: {
      requiresAuth: true
    }
  },
  {
    path: '/my-offers',
    name: 'MyOffers',

        component: MyOffers,
        meta: {
          requiresAuth: true
        }
      },
      {
        path: '/messages',
        name: 'Messages',
        component: Chats,
        meta: {
          requiresAuth: true
        }
      }

  ];
  const index = createRouter({
    history: createWebHistory(),
    routes,
    linkActiveClass: "active"
  });

  index.beforeEach((to, from, next) => {
    const isLoggedIn = !!localStorage.getItem('access_token');
```

```
if (to.meta.requiresAuth && !isLoggedIn) {
  next({ path: '/login' });
} else if (to.meta.guest && isLoggedIn) {
  next({ path: '/login' });
} else {
  next();
}

});

export default index;
```

Home.vue

Компонент головної сторінки

```
<template>
  <div class="banner">
    <div class="bgDark"></div>
    <div class="container h-100 z-2 position-relative">
      <div class="bg-dark-gray h-100 color-white d-flex flex-column justify-content-center align-items-center">
        <h1 class="text-center">Ласкаво просимо до BookSharing</h1>
        <p class="lead text-center mt-4">
          BookSharing – книжкове джерело, що об'єднує книголюбів! Це простір для обміну літературними шедеврами, досвідом та думками. З нами кожен читач має можливість відкривати нові горизонти та ділитися своїм літературними враженнями та читацьким досвідом.
        </p>
<!--        <p class="text-center mt-1">
          Ми віримо, що кожна книга має свою історію, але часом найцікавіші історії виникають, коли книга міняє свого читача.
        </p>-->
        <router-link to="/books" class="btn btn-outline-light">Перейти до списку книг</router-link>
      </div>
    </div>
</div>
```

```
<div class="container">
    <div class="row">
        <div class="col-12 text-center mx-4 my-4">
            <h2 class="pageTitle">Останні книги</h2>
        </div>
    </div>
    <div class="row">
        <Book
            v-for="book in books"
            :key="book.id"
            :book="book" />
    </div>
  </div>
</template>

<script>
import Book from "../Blocks/Book.vue";

export default {
    name: "Home",
    components: {Book},
    data() {
        return {
            books: [],
        };
    },
    created() {
        this.fetchBooks();
    },
    methods: {
  fetchBooks() {
      axios.get('/api/books/last')
          .then(response => {
              this.books = response.data;
          })
```

```
            .catch(error => {
                console.error('There was an error fetching the books:', error);
            });
        },
    }
};
</script>

<style>

 .banner {
    background: url('/images/banner.jpg') no-repeat center center;
    background-size: cover;
    height: 700px;
    position: relative;
 }

 .bgDark {
    background: rgba(0, 0, 0, 0.5);
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: 1;
 }

 .color-white {
    color: white;
 }
</style>
```

BookList.vue

Компонент сторінки з книгами

```
    <template>
        <div class="container mt-4">
            <div class="row">
```

```
        <div class="col-12">
          <h2 class="pageTitle">Книги</h2>
        </div>
      </div>
      <div class="row">
        <Book
          v-for="book in books.data"
          :key="book.id"
          :book="book" />
      </div>
      <!-- Pagination -->
      <nav aria-label="Page navigation">
        <ul class="pagination">

          <ul class="pagination">
            <li class="page-item" v-if="books.prev_page_url">
              <a class="page-link" href="#" aria-label="Previous" @click="fetchBooks(books.current_page -
1)">
                <span aria-hidden="true">&laquo;</span>
              </a>
            </li>
            <li class="page-item" v-for="page in books.last_page" :key="page" :class="{'active': page ===
books.current_page}">
              <a class="page-link" href="#" @click="fetchBooks(page)">{{ page }}</a>
            </li>
            <li class="page-item" v-if="books.next_page_url">
              <a class="page-link" href="#" aria-label="Next" @click="fetchBooks(books.current_page +
1)">
                <span aria-hidden="true">&raquo;</span>
              </a>
            </li>
          </ul>

        </nav>
      </div>
    </template>
```

```
<script>
import Book from "../../Blocks/Book.vue";

export default {
    name: "BookList",
    components: {Book},
    data() {
        return {
            books: {
                data: [],
                current_page: null,
                last_page: null,
                next_page_url: null,
                prev_page_url: null
            },
        };
    },
    created() {
        this.fetchBooks(1);
},
methods: {
    fetchBooks(page) {
        axios.get(`/api/books?page=${page}`)
            .then(response => {
                this.books = response.data;
            })
            .catch(error => {
                console.error('There was an error fetching the books:', error);


            });
    },
}
};
</script>
```

BookDetail.vue

Компонент сторінки книги

```
<template>
  <div class="container mt-4">
    <div class="row shadow rounded p-4">
      <div class="col-12">


        <div v-if="loaded" class="book-detail">
          <div class="book-image">
            <img :src="book.image" alt="Book Image">
          </div>

          <div class="book-info">
            <h1>{{ book.title }}</h1>
            <p class="category-genre">{{ book.category.name }} / {{ book.genre.name }}</p>
            <p class="author">
              Автор: {{ book.author.name }} <br>
              Рік: {{ book.year }} <br>
              ISBN: {{ book.isbn }} <br>
              Мова: {{ book.language }} <br>
              Видавництво: {{ book.publisher }} <br>
              Сторінок: {{ book.pages }}
            </p>
            <p class="description">{{ book.description }}</p>



          </div>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-12">
        <h2 class="pageTitle">Примірники</h2>
```

```
        </div>
      </div>
      <div class="row shadow rounded p-4">
        <div class="col-12">
          <div class="exchange-offers">
            <ul v-if="book.instance && book.instance.length">
              <li v-for="instance in book.instance">
                <button class="btn btn-outline-primary btn-sm"
@click="makeExchangeOffer(instance.id)">Запропонувати обмін</button> <span>{{
instance.current.name }} (ID: {{ instance.id }})</span>
              </li>
            </ul>
            <p v-else>
              Примірників не знайдено
            </p>
          </div>

        </div>
  <div v-if="createOffer" class="row">
    <div class="col-12 col-md-12 border bg-white">
      <p>Оберіть книгу для обміну</p>
      <div v-for="item in myBooks">
        <input type="radio" :id="item.id" :value="item.id" v-model="selectedBook">
        <label :for="item.id">{{ item.book.title }}</label>
      </div>
      <button class="btn btn-primary my-2" @click="createExchangeOffer">Підтвердити</button>
    </div>
  </div>
</div>

<div class="container">
  <div class="row">
```

```
        <div class="col-12">
          <h2 class="pageTitle">Інші книги цього жанру</h2>
        </div>
      </div>
      <div class="row">
        <Book
          v-for="item in genreBooks"
          :key="item.id"
          :book="item" />
      </div>
    </div>
</template>

<script>
import Book from "../../Blocks/Book.vue";
import { useToast } from "vue-toastification";

export default {
  components: {Book},
  props: {
    id: {
      type: Number,
      required: true,
    },
  },
  data() {
    return {
      book: [],
      genreBooks: [],
      myBooks: [],
      selectedBook: null,
```

```
        selectedInstanceToReceive: null,
        loaded: false,
        createOffer: false,
        toast: useToast(),
    };
},
created() {
    this.fetchBookDetails();
},

watch: {
    '$route.params.id': {
        immediate: true,
        handler(newId) {
            window.scrollTo(0, 0);
            this.fetchBookDetails();
        }
    }
},
  methods: {
      fetchBookDetails() {
          axios.get(`/api/books/${this.id}`)
              .then(response => {
                  this.book = response.data;
                  this.fetchGenreBooks();
                  this.loaded = true;

              })
              .catch(error => {
                  console.error('Error fetching book details:', error);
              });
      },
```

```
    fetchMyBooks() {
      axios.get('/api/books/my')
        .then(response => {
          this.myBooks = response.data;
        })
        .catch(error => {
          console.error('Error fetching my books:', error);
        });
    },
    fetchGenreBooks() {
      axios.get(`/api/books/genre/${this.book.genre_id}`)
        .then(response => {
          this.genreBooks = response.data;
        })
        .catch(error => {
          console.error('Error fetching genre books:', error);
        });
    },
  makeExchangeOffer(selectedInstanceToReceive) {
    this.fetchMyBooks();
    this.selectedInstanceToReceive = selectedInstanceToReceive;
    this.createOffer = true;
  },
  createExchangeOffer() {
    axios.post('/api/exchange-offers', {
      give_book_instance_id: this.selectedBook,
      take_book_instance_id: this.selectedInstanceToReceive,
    })
      .then(response => {
        this.createOffer = false;
        this.toast.success('Пропозиція обміну надіслана');
      })
```

```
            .catch(error => {
                console.error('Error creating exchange offer:', error);
            });
        },
    },
};
</script>

<style scoped>
.book-detail {
    display: flex;
}

.book-image img {
    max-width: 200px;
}

.book-info {
    margin-left: 20px;
}

.category-genre {
    color: #666;
    font-style: italic;
    /* Add more styling */
}

.description {
    margin-top: 10px;
    /* Add more styling */
}
```

```
.exchange-offers {
    margin-top: 20px;
    /* Add more styling */
}

/* Add more styling as needed */
</style>
```