Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Веб-додаток «Book Exchange» для пошуку книг та обміну ними
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПІПЗ.190150.19.05.ПЗ

Виконав студент III курсу група ІПЗс-19-1 _____ О.О.Кирилюк
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ Н. І. Праворська
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Бедратюк
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

1 червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
01 03 2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Кирилюк Ользі Олександрівні
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Веб-додаток "Book Exchange" для пошуку книг та обміну ними

Керівник проекту (роботи) Праворська Наталія Іванівна
кандидат педагогічних наук, доцент

Затверджена наказом ректора університету від 01.03.2022 р. № 18

2. Строк подання студентом проекту (роботи) на кафедру 07.06.2022 р.

3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмноїсистеми

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди, 21 шт.)

6. Консультанти розділів дипломного проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Бездратюк Г.І., ст. викладач кафедри ІПЗ		
Антиплагіат	Гурман У.В., доцент кафедри ІПЗ		

7. Дата видачі завдання « 01 » березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем ДП	01.12– 30.12.2021	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2022	
3 Проектування програмного забезпечення	01.02 – 28.02 2022	
4 Програмна реалізація	01.03 – 10.04.2022	
5 Тестування програмного забезпечення	11.04 – 30.04.2022	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2022	
7 Попередній захист ДП	травень 2022 (згідно графіка)	
8 Перевірка ДП на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2022	
9 Підготовка до захисту та захист ДП	з 01.06.2022	

Студент _____ О.О.Кирилюк
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Н. І. Праворська
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: Веб-додаток «Book Exchange» для пошуку книг та обміну ними.

Автор проекту: Кирилюк Ольга Олександрівна.

Керівник проекту: Праворська Наталія Іванівна.

Пояснювальна записка: 103 с., 41 рис., 5 табл., 4 дод., 15 джерел.

Графічна частина: 21 презентаційний слайд.

HTTP, MVC, UML, ПЗ, GDI, CLI.

Мета проекту – розробити веб-додаток для пошуку та обмінукнигамиза допомогою мережі Інтернет, застосувати знання та вміння, отримані у процесі навчання.

У дипломному проекті проведено аналіз предметної області, визначені функціональні та нефункціональні вимоги до програмної системи, розроблена архітектура додатку, спроектована база даних, виконано програмну реалізацію проекту та проведено тестування.

Для реалізації програмногозабезпеченнявикористано такі технології:

- мова програмування PHP;
- PHP-фреймворк Laravel;
- HTML, CSS, Bootstrap, Javascript, JQuery.

Збереження даних відбувається за допомогою веб-застосунку для адміністрування бази даних - phpMyAdmin.

1 червня 2022 р.
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПІПЗ.190150.19.05.ПЗ	Пояснювальна записка	103		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні	21		
			слайди			

Змн.	Арк.	№ докум.	Підпис	Дата	ДПІПЗ.190150.19.05.ВД		
Виконав		Кирилюк О.О.		1.06	Веб-додаток "BookExchange" для пошуку книг та обміну ними Відомість документів	Літ.	Арк. 1 / Аркушів 1
Керівник		Праворська Н.		1.06			
Н. контр.		Бедратюк Г. І.		1.06		ХНУ, ІПЗс-19-1	
Зав. каф.		Бедратюк Л.П.		1.06			

ЗМІСТ

ВСТУП

В сучасному технологічному житті Інтернет відіграє важливу роль в житті кожної людини. Більше того, Інтернет – це винахід передової науки та сучасних технологій.

Із значним прогресом технологій значення Інтернету з часом лише зросло. Залежність від Інтернету пов’язана з багатьма перевагами, які він може запропонувати – наприклад, зниження робочого стресу та зміна зовнішнього вигляду спілкування.

Використовуючи Інтернет, з’являється можливість знайти різноманітну інформацію про світ. В Інтернеті розміщена Вікіпедія, яка вважається однією з найбільших найкращих довідників, яку веде велика спільнота вчених-добровольців і редакторів з усього світу. Через Інтернет можна отримати відповіді на всю свою цікавість. У сфері освіти – це також відіграє важливу роль, особливо враховуючи пандемію та війну. Інтернет під час пандемії став легкою альтернативою для заміни традиційної системи освіти та пропонує додаткові ресурси для дистанційних занять.

Наявність Інтернету поступово замінює використання традиційних газет. Він також пропонує різноманітні рекреаційні переваги. Можна сказати, що Інтернет відіграє велику роль у підвищенні якості життя.

Попри те, що сучасний світ з кожним роком стає все більш модернізованим і з’являються все нові можливості для зручного читання книг в електронному форматі, паперова книга й досівважається кращим способом занурення у чарівний світ для книгоманів.

А за допомогою програмнихпродуктів з’являються нові способи пошуку потрібної книги (особливо, якщо вона застаріла та її важко знайти на полицях в магазині) або обміну книги для тих, хто не має можливості дозволити собі купувати нові книги. Адже книга дозволяє розвивати аналітичні здібності, покращувати пам’ять, стимулювати мозок. А також книга є своєрідною

розвагою, оскільки існує велика різноманітність сюжетів та жанрів. З книгами джерело розваг не завершується ніколи. І воно завжди доступне.

Саме тому було обрано дану предметну область для реалізації дипломного проекту.

Основні завдання:

– дослідити предметну область;
– проаналізувати існуючі подібні рішення;
– визначити функціональні та не функціональні вимоги;
– спроектувати структуру;
– розробити базу даних;
– створити UML діаграми;
– розробити макет та дизайн веб-сайту;
– розробити програмне забезпечення;
– протестувати проект на коректність роботи.

Після завершення останнього завдання, результатом дипломного проекту виступає функціонуюча система, яка надає користувачу можливість обмінюватися книгами з будь-якого пристрою за допомогою мережі Інтернет.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Темою роботи є розробка веб-додатку для зручного пошуку книг та обміну ними.

Мета розробки дипломного проекту – реалізувати веб-додаток з можливістю обмінюватися книгами з іншими користувачами.

Актуальність теми полягає в тому, що завдяки розробці даного програмного продукту користувач має змогу зекономити кошти, що є великою перевагою, особливо останнім часом, в період війни.

Веб-додаток – це комп'ютерна програма, яка використовує веб-браузер для виконання певної функції.

Веб-додаток є клієнт-серверною програмою. Це означає, що він має клієнтську і серверну сторону. Термін «клієнт» тут відноситься до програми, яку особа використовує для запуску програми. Це частина клієнт-серверного середовища, де багато комп'ютерів обмінюються інформацією. Наприклад, у випадку з базою даних, клієнтом є програма, за допомогою якої користувач вводить дані. Сервер – це програма, яка зберігає інформацію.

Система онлайн-обміну книг – це веб-портал, за допомогою якого з'являється можливість обмінюватися книгами в межах спільноти. Будь-який користувач може використовує даний сервіс, для цього потрібно лише зареєструватися.

Користувач також може подати онлайн-запит для обміну книгами. Ця система в основному призначена для того, щоб допомогти читачам, в яких немає змоги систематично купляти нові книги. При використанні даної системи, користувач може обмінюватися власною книгою з іншими користувачами, та читати їх книги.

Переваги використання веб-сайту від традиційного буккросингу:

– доступ до програмного продукту в будь-який момент часу, на будь-якій території;

– швидкий пошук будь-якої книги завдяки зручному пошуку на сторінці;

– економія коштів;

– безпечність використання.

Недоліками використання веб-сайту є реєстрація та авторизація, яка займає певний час, проте дозволяє ідентифікувати користувачів.

Процес обміну книгами зазвичай відбувається таким шляхом:

– реєстрація користувачем;

– додання власної книги для обміну;

– знаходження книги;

– відправлення запиту на обмін;

– при наявності книги і бажанні власника (при умові, що власник знайшов книгу для обміну) відбувається обмін книгами;

– після прочитання книг користувачі повертають книги один одному.

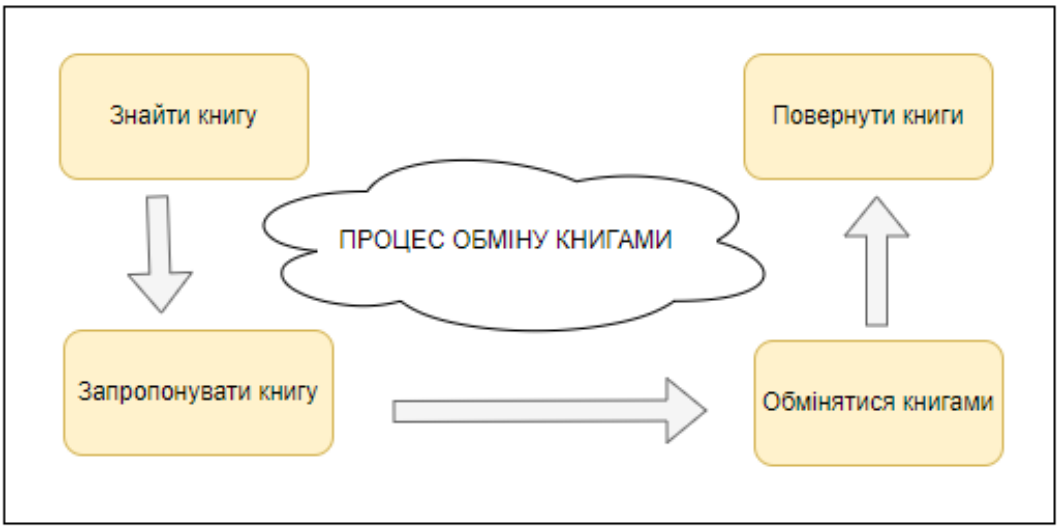


Рисунок 1.1 – Процес обміну книгами

На рисунку 1.1 зображено процес обміну книгами, який реалізовуватиметься в веб-додатку. За допомогою даного процесу з’являється можливість легко та просто реалізувати структуру додатку.

Отже, проаналізувавши предметну область, переходимо до аналізу наявного програмно-технічного забезпечення, яке за предметною областю є подібне до теми дипломного проекту.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Для якісної розробки програмного забезпечення необхідно проаналізувати існуючі програмні продукти, предметна область яких збігається з областю теми дипломного проекту. Для кожного програмного продукту слід визначити пункти, наведені нижче:

– переглянути веб-сайт існуючого рішення;

– визначити основні моменти, зручні для користувача;

– проаналізувати переваги та недоліки;

– обрати для власного продукту пункти, яких не вистачає в реальних проектах, або ж функції, які необхідно вдосконалити.

Нижче наведено найпопулярніші сайти обміну книгами.

На рисунку 1.2 зображений веб-сайт BookMooch.

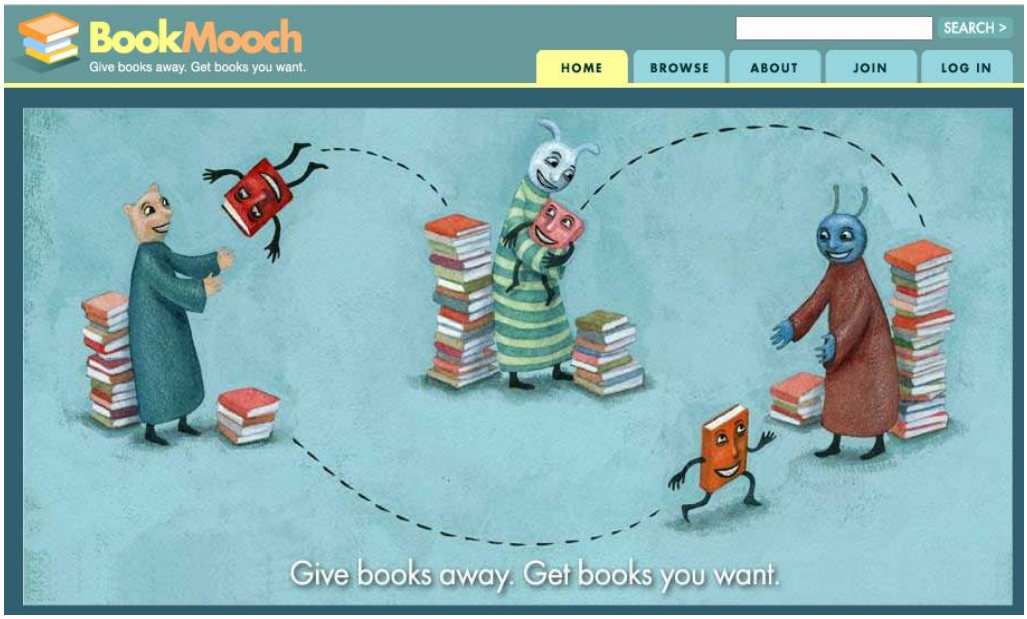


Рисунок 1.2 – BookMooch

Працює веб-сайт BookMooch за системою балів – користувач отримує 1/10 бала за кожну книгу, яку додає до свого інвентарю. Замовлення (запит) на книгу коштує один бал за книги з країни користувача, два бали – за міжнародні. Надсилаючи книгу, кожен отримує один бал за національні обміни і три бали за міжнародні. Це надає можливість створити список побажань, а сайт надсилає електронний лист, коли книга з списку побажань буде доступна.

Переваги:

– сайт безкоштовний;

– цікаві зображення, концепція веб-сайту;

– приємне поєднання кольорів на сайті;

– зручна навігація веб-продукту.

Недоліки:

– проблеми з адаптивністю сайту;

– помилки при переході на певні сторінки сайту;

– погано реалізований пошук книг;

– відсутня фільтрація книг за жанрами.

На рисунку 1.3 зображений веб-сайт PaperBackSwap.

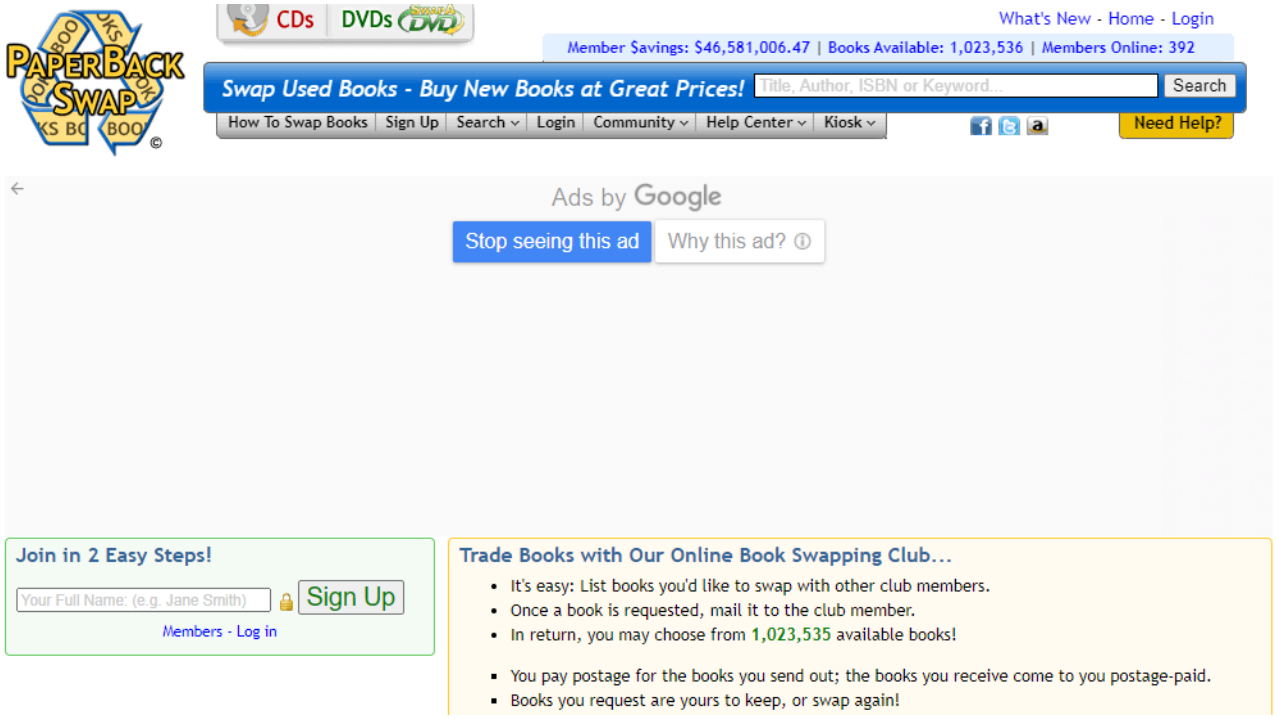


Рисунок 1.3 – PaperBackSwap

В даному веб-додатку існує система балів. Сайт є безкоштовним, але залишає за собою право розпочати стягнення плати в майбутньому, відповідно до їх поширених запитань. Учасники також можуть придбати бали.

Існує варіант списку побажань, але якщо перед користувачем в черзі є учасники, які також мають книгу в списку бажань, вони отримають перші результати.

Веб-сайт PaperBackSwap надає користувачам можливість друкувати поштові відправлення з сайту.

Переваги:

– великий каталог книг;

– реалізований пошук книг;

– реалізована реєстрація, авторизація.

Недоліки:

– велика кількість реклами;

– надто простий дизайн сайту в плані вигляду;

– важко орієнтуватися на сайті;

– проблема в реалізації реєстрації/авторизації;

– відсутність інструкції користування сайтом.

На рисунку 1.4 зображений веб-сайт BookSwap. Веб-додаток працює без системи балів – користувач просто переглядає інвентар і запитує потрібні книги.

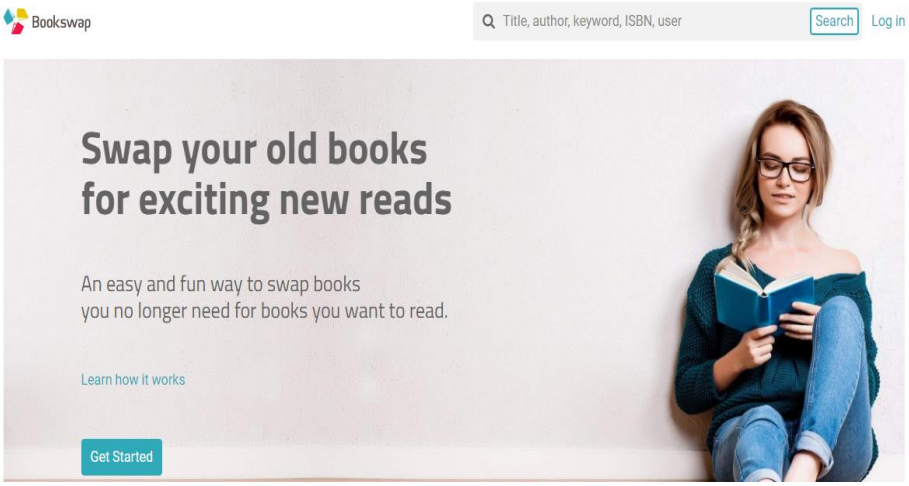


Рисунок 1.4 – BookSwap

Одержувач оплачує поштові витрати, а відправник друкує етикетку з оплаченою поштою з сайту, загортає книгу та надсилає її. Сайт «безкоштовний», але користувач повинен додати десять доларів США до свого облікового запису, перш ніж запитувати книги. На сайті присутня функція списку бажань.

Переваги:

– приємний на вигляд дизайн;

– зручний у використанні веб-сайт.

Недоліком веб-сайту BookSwap виступає постійна присутність реклами.

На рисунку 1.5 зображено веб-сайт Readitswapit.

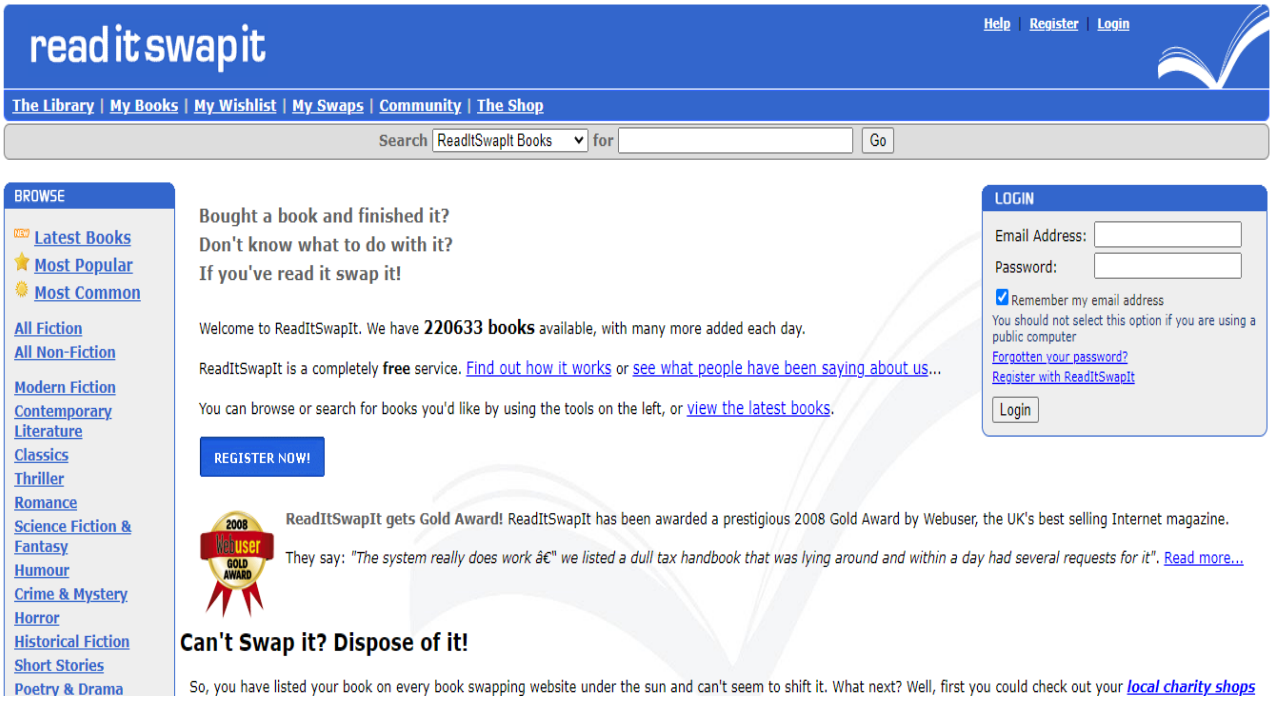


Рисунок 1.5 – Readitswapit

Алгоритмом роботи виступає пряма заміна – коли користувач запитує у когось книгу, обмін відбувається лише в тому випадку, якщо вони знайдуть у інвентарі щось потрібне (це запобігає накопиченню балів через те, що користувачі не можуть знайти потрібну книгу на сайті).

Якщо користувача книги не цікавлять, обмін відхиляється. Кількість книг, які користувач може замовити, обмежена відгуками.

Перевагою веб-сайту Readitswapit являється зручна навігація по сайту та велика кількість книг в асортименті.

Недоліком – надто простий дизайн сайту в плані вигляду.

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання

Детальний аналіз предметної області, її структурних та функціональних особливостей, а також аналіз наявного програмно-технічного забезпечення дозволяє сформувати функціональні та не функціональні вимоги до програмного забезпечення, яке розроблятиметься.

Функціональні вимоги до програмного забезпечення:

– реєстрація;
– авторизація;
– перегляд всіх книг;
– пошук книг за жанром;
– пошук книг за автором;
– пошук книг за назвою;
– перегляд однієї книги;
– надсилання запиту на книгу;
– перегляд запитів на обмін;
– прийняття запиту;
– відхилення запиту;
– відправлення повідомлення на пошту;
– повернення книги;
– вихід з системи.

Не функціональні вимоги до програмного забезпечення:

– зручний інтерфейс;

– легкість у використанні;

– автентифікація доступу;

– адаптованість до мобільних пристроїв.

Для того, щоб детально оглянути вимоги програмного забезпечення використовується мова візуального моделюванняUML.

Діаграма UML – це діаграма, заснована на UML (Unified Modeling Language) з метою візуального представлення системи разом з її основними акторами, ролями, діями, артефактами або класами, щоб краще зрозуміти, змінити, підтримувати або документувати інформацію про систему.

Вона заснована на схематичних представленнях програмних компонентів. Використовуючи візуальні уявлення, краще зрозуміти можливі недоліки або помилки в програмному забезпеченні або бізнес-процесах.

Складемо описи користувачів системи (акторів) та необхідного їм функціоналу та сервісів (варіанти використання).

У таблиці 1.1 наведені актори розроблюваного програмного комплексу.

Таблиця 1.1 – Опис акторів розроблюваного ПЗ

Актор	Короткий опис
Незареєстрований користувач (гість)	Ввійшовши на сайт може перейти на сторінки реєстрації та авторизації, а також переглядати основну інформацію про книги без можливості пошуку та обміну.
Зареєстрований користувач	Може додавати власні книги, переглядати кнги інших користувачів і також пропонувати обмін.

В таблиці 1.2 наведені варіанти використання розроблюваного ПЗ.

Таблиця 1.2 – Опис варіантів використання розроблюваного ПЗ

Актор	Найменування ВВ	Опис ВВ
1	2	3
Гість	Реєстрація	Користувач має можливість зареєструватися в системі.
	Перегляд запропонованих книг	Користувач може переглядати книги, проте без можливості обміну та пошуку.
Зареєстрований користувач	Авторизація	Зареєстрований попередньо користувач, має можливість ввійти в систему.
	Створення книги	Користувач може додати книги, якими згоден ділитися зі світом.

Продовження таблиці 1.2

	Перегляд всіх книг	Користувач може переглядати всі книги, існуючі в бібліотеці веб-сайту.
	Перегляд книг за певним жанром	Користувач переглядатиме книги лише того жанру, який його цікавить.
	Пошук книги	Користувач має можливість шукати певну книгу.
	Обмін	При бажанні іншого користувача відправляються на пошту додаткові данні стосовно обміну.

Визначивши акторів системи та варіанти використання, побудуємо діаграму варіантів використання (рисунок 1.6).

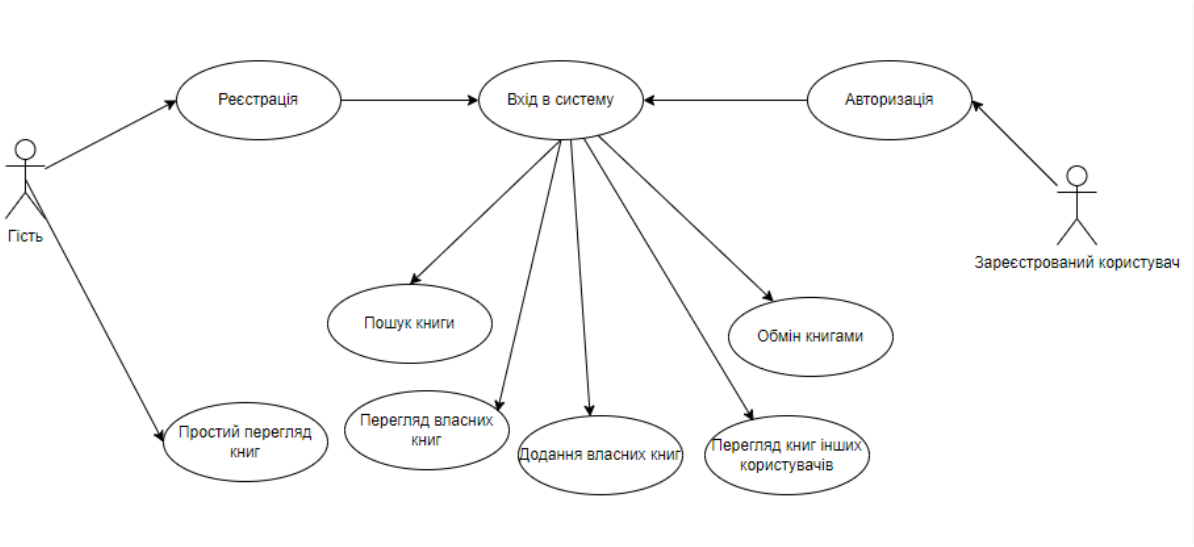


Рисунок 1.6 – Діаграма варіантів використання розроблюваного ПЗ

При вході в систему користувач міг бути попередньо зареєстрованим, в цьому випадку йому потрібно лише пройти авторизацію, якщо користувач не реєструвався, йому необхідно пройти дану процедуру для подальшої роботи з сайтом.

Після входу в систему користувач може вільно працювати з усіма функціями, що надані в системі.

Після завершення аналізу вимог до програмного забезпечення було розроблене технічне завдання, яке подано у додатку А.

У даному розділі було проведено аналіз та огляд предметної області, розглянуто існуючі рішення та проблеми, а також були визначені функціональні вимоги до розроблюваного ПЗ.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕНЯ

2.1 Аналіз та проектування архітектури системи

Клієнт-серверна архітектура – це обчислювальна модель, в якій сервер розміщує, забезпечує і керує більшістю ресурсів і послуг, які споживає клієнт. Даний тип архітектури має один або кілька клієнтських комп'ютерів, підключених до центрального сервера через мережу, або підключення до Інтернету.

Модель даної системи працює наступним чином:

1. Клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно.

2. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше. На рисунку 2.1 зображена схема архітектури:

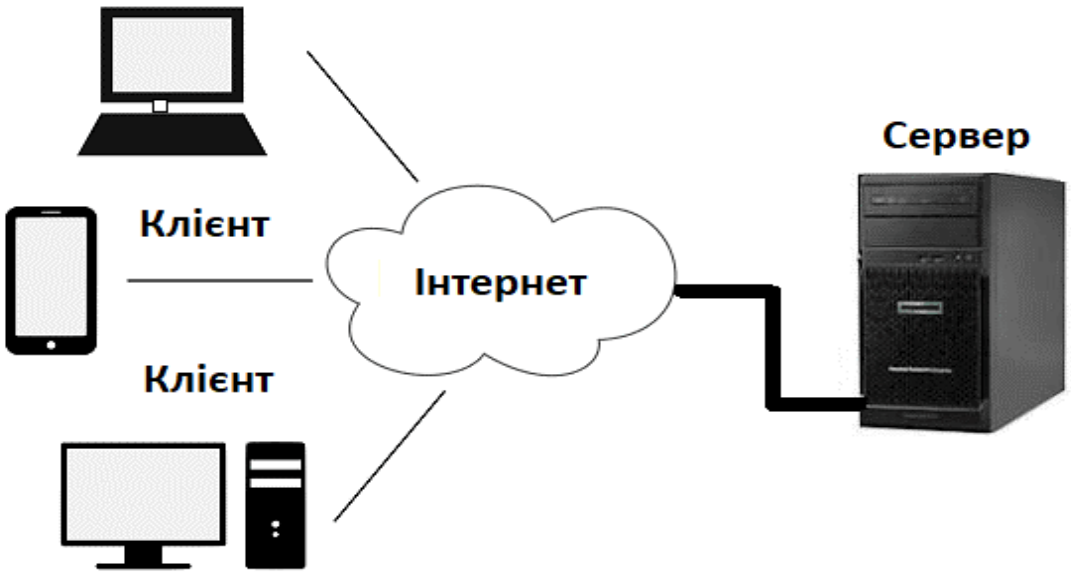


Рис. 2.1 – Схема клієнт-серверної архітектури

Існує чотири різні типи архітектури клієнт-сервер.

– архітектура 1 рівня;

– 2-рівнева архітектура;

– 3-рівнева архітектура;

– архітектура рівня N.

Протокол передачі гіпертексту (HTTP) є основою всесвітньої павутини, який використовується для завантаження веб-сторінок за допомогою гіпертекстових посилань.

HTTP – це протокол прикладного рівня, призначений для передачі інформації між мережевими пристроями і працює поверх інших рівнів стеку мережевих протоколів. Типовий потік через HTTP передбачає, що клієнтська машина робить запит до сервера, який потім надсилає повідомлення-відповідь.

Нижче розглянуто основні HTTP-методи для HTTP-запиту:

– GET –метод, що використовується для отримання ресурсу;

– POST –метод, який використовується для додавання нового ресурсу;

– PUT –метод, призначений для оновлення ресурсу;

– PATCH –метод, що дозволяє виконати часткове оновлення ресурсу;

– DELETE –метод, що використовується для видалення ресурсу;

– OPTIONS–метод, необхідний для опису параметрів з’єднання з обраним ресурсом на сервері.

Дотримання вказаного архітектурного стилю дозволить створити систему, яку буде легко оновлювати та масштабувати, змінювати без вимикання системи.

2.2 Аналіз, вибір та проектування архітектури веб-додатку

Використовуючи архітектурний партерн MVC передбачає те, що система містить компонети: M – моделі (models), C – контролери (controllers), V – представлення (views).

Що означає кожен із цих компонентів:

1. Модель: бекенд, який містить всю логіку даних.

2. Перегляд: інтерфейс або графічний інтерфейс користувача (GUI).

3. Контролер: мозок програми, який контролює відображення даних.

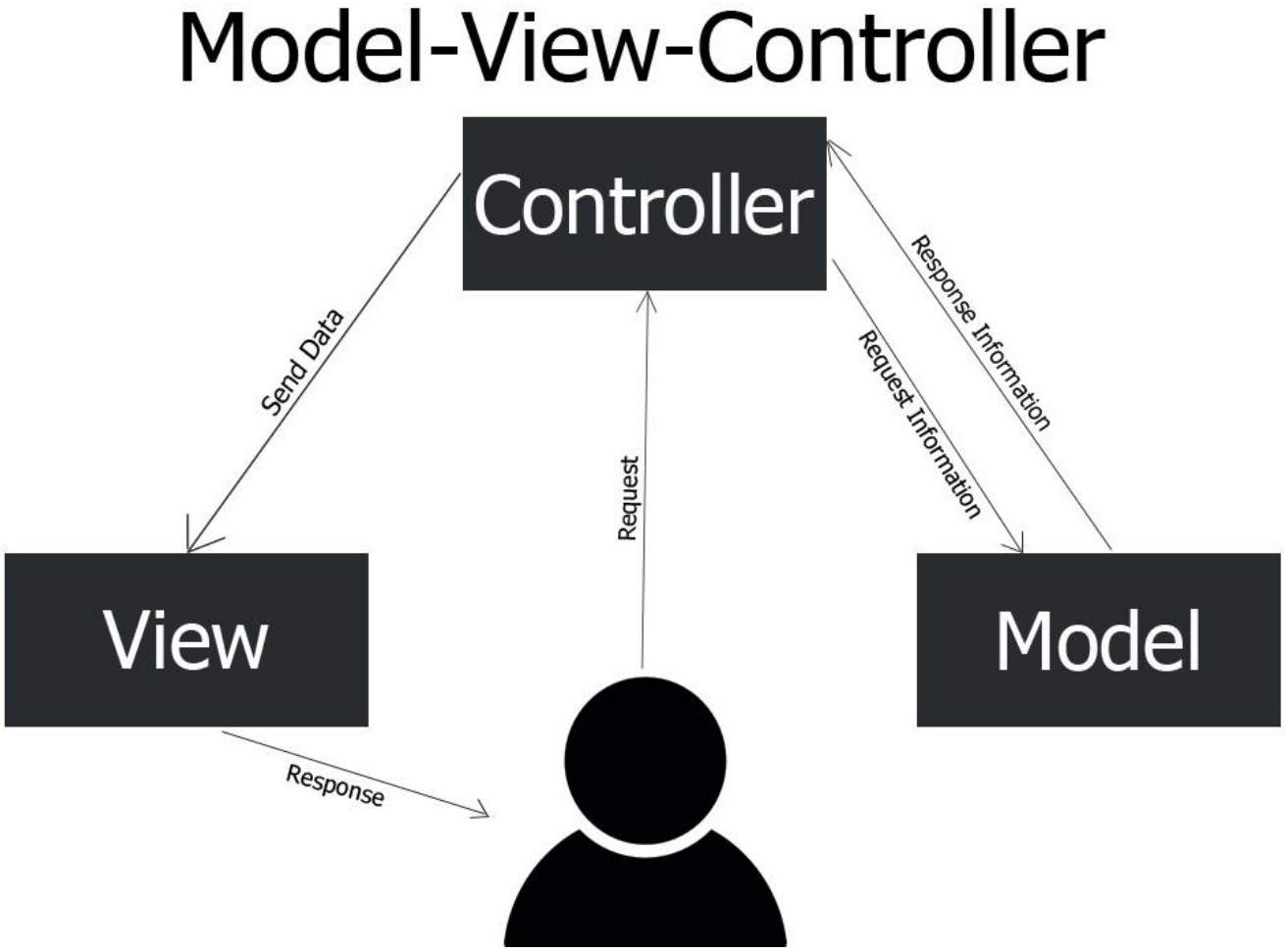


Рисунок 2.2 – Схема роботи MVC

Концепцію MVC вперше представив Трюгве Реєнскауг, який запропонував її, як спосіб розробки графічних інтерфейсів настільних додатків.

Сьогодні шаблон MVC використовується для сучасних веб-додатків, оскільки. він дозволяє програмі бути масштабованою, підтримуватися та легко розширюватися.

Шаблон MVC допомагає розбити інтерфейсний і бекенд-код на окремі компоненти. Таким чином, набагато легше керувати та вносити зміни до будь-якої сторони, не заважаючи одна одній.

Зробити зазначене досить складно, особливо коли кільком розробникам потрібно оновити, змінити або налагодити повноцінну програму одночасно.

Найпривабливішою концепцією шаблону MVC є розділення проблем.

Сучасні веб-додатки дуже складні, і внесення змін, іноді може бути великим головним болем.

Керування інтерфейсом і бекендом у менших окремих компонентах дозволяє програмі бути масштабованою, підтримуваною та легко розширюваною.

Переваги використання даного шаблону:

– декілька розробників можуть одночасно працювати над моделлю, контролером і представленнями;

– MVC дозволяє логічно групувати пов’язані дії на контролері разом. Погляди для конкретної моделі також групуються разом;

– моделі можуть мати кілька видів.

Недоліки використання MVC:

– навігація по фреймворку може бути складною, оскільки вона вводить нові шари абстракції та вимагає від користувачів адаптації до критеріїв декомпозиції MVC;

– знання про декілька технологій стають нормою. Розробники, які використовують MVC, повинні володіти кількома технологіями.

2.3 Аналіз та вибір типу бази даних, проектування структури бази даних

2.3.1 Логічна модель бази даних

Логічна модель описує основні сутності системи та зв’язки між ними.

Сутність User необхідна для реєстрації користувача та необхідної інформації для подальшої роботи з системою.

Сутність Genre потрібна для збереження даних стосовно жанрів книг, які можуть існувати.

Сутність Book використовується безпосередньо для створення книг.

Сутність Exchange необхідна для збереження даних стосовно обміну книгами між користувачами.

Схема бази даних зображенана рисунку 2.3:

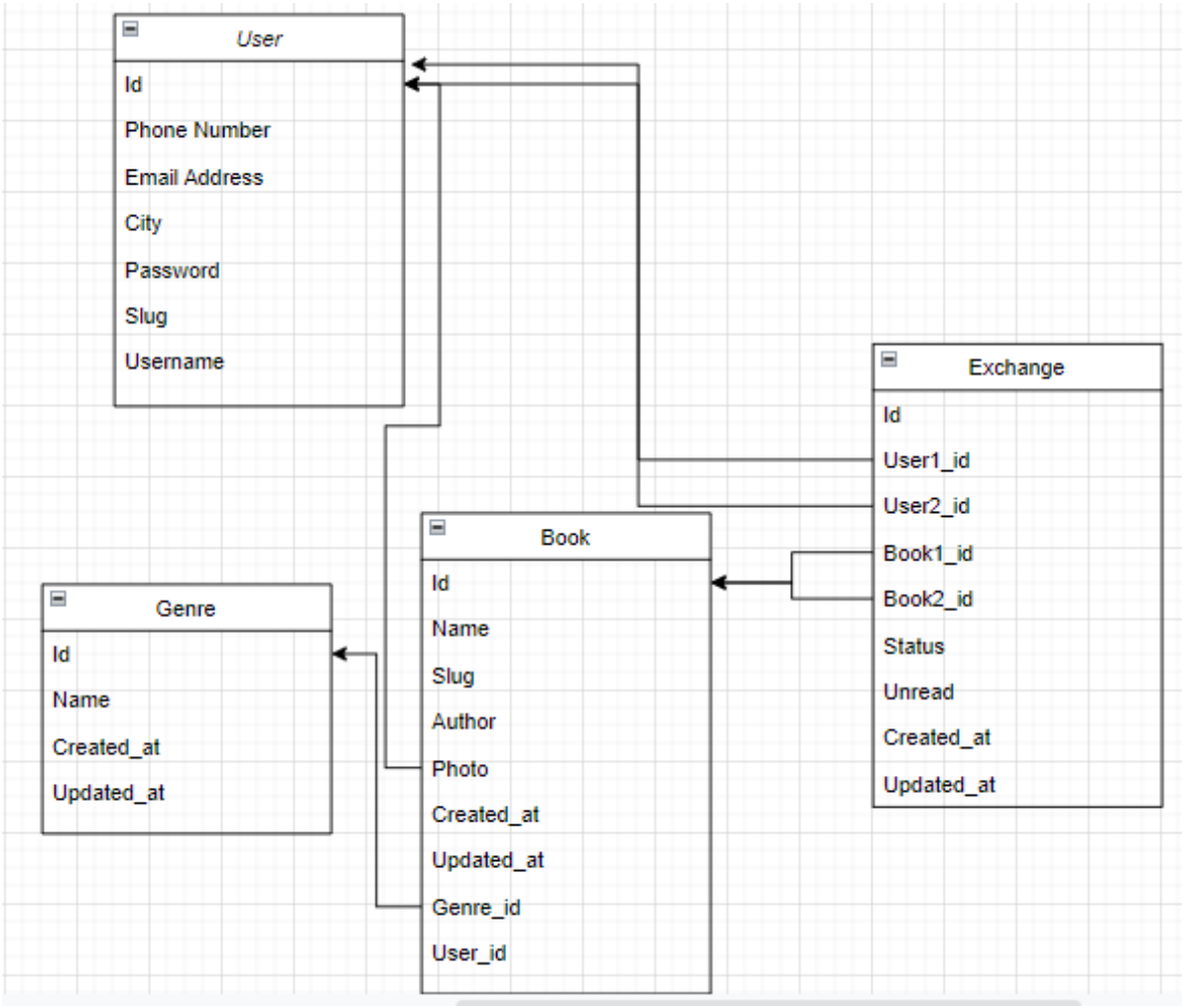


Рисунок 2.3–Схема бази даних

2.3.2 Фізична модель бази даних

Фізична модель бази даних створюється на основі логічної моделі. Це модель даних, реалізована у вигляді опису бази даних. Вона відображає спосіб розміщення даних в середовищі зберігання та спосіб доступу до даних.

Для сховища даних будуть використовуватися phpMyAdmin – це інструмент, написаний на PHP, який дозволяє адмініструвати базу даних MySQL через інтерфейс браузера (форми, кнопки, посилання).

2.3.3 Опис таблиць веб–додатку

а) Users – таблиця, призначена для реєстрації користувачів, складається з таких полів:

1. id;
2. name;
3. slug;
4. city;
5. pnonenumber;
6. email;
7. password;
8. created_at;
9. updated_at.

б) Genre – таблиця, що містить дані стосовно жанрів книг:

1. id;
2. name;
3. created_at;
4. updated_at.

в) Book – таблиця, призначена для збереження даних про книги:

1. id;
2. name;
3. slug;
4. author;
5. photo;
6. created_at;

7. updated_at;
8. genre_id;
9. user_id.

г) Exchange – таблиця, що містить дані стосовно обміну книгами:

1. id;
2. user1_id;
3. user2_id;
4. book1_id;
5. book2_id;
6. status;
7. unread;
8. created_at;
9. updated_at.

2.4 Проектування інтерфейсу користувача

Основна сторінка при вході на сайт зображена на рисунку 2.4:

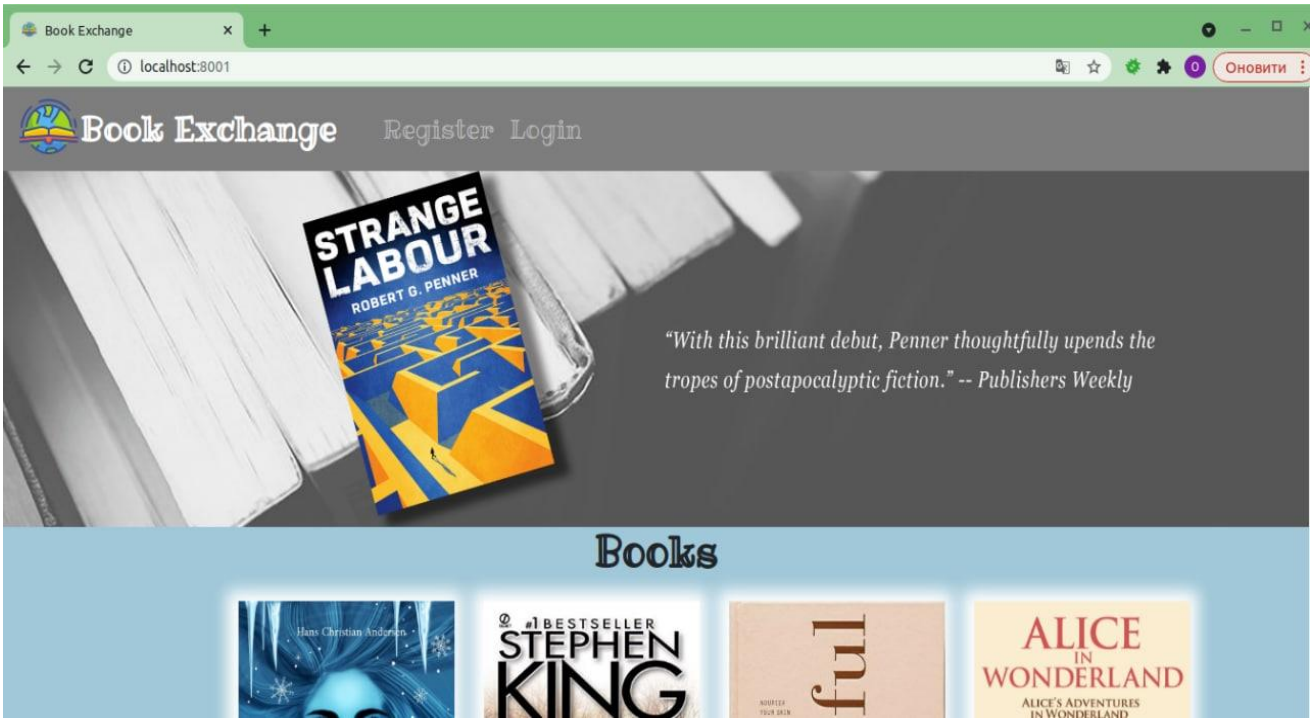


Рисунок 2.4 – Перша сторінка сайту

На сторінці можна побачити меню, яке складається з двох пунктів: реєстрації та авторизації.

Також тут відображається картинка-банер та перелік книг.Кожна картинка книги містить посилання на сторінку, за допомогою якої користувач може дізнатися деталі книги.

Друга частина сторінки зображена на рисунку 2.5:

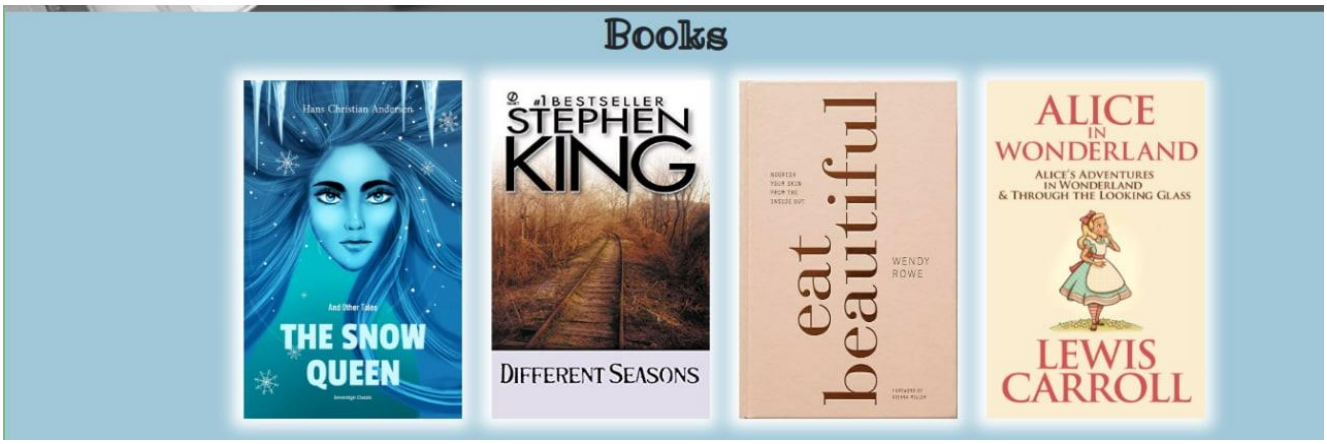


Рисунок 2.5 – Друга частина першої сторінки сайту

Дана сторінка призначена для гостей, або неавторизованих користувачів з метою ознайомлення з веб-сайтом.

При переході на сторінку авторизації в меню відображається сторінка, зображена на рисунку 2.6.

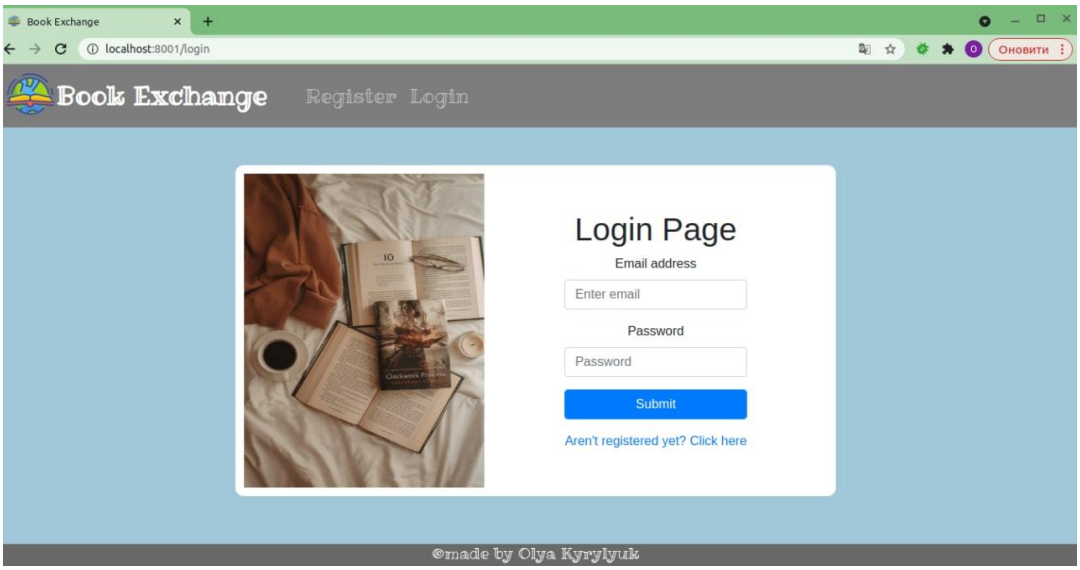


Рисунок 2.6 – Сторінка авторизації веб-сайту

Сторінка містить два поля, які необхідні для аутентифікації особи. Якщо користувач випадково зайшов на сторінку авторизації, він може перейти за посиланням під кнопкою та перейти на реєстрацію, або ж виконати ту ж саму дію натиснувши на відповідне поле в меню.

Реєстрація веб-додатку зображена на рисунку 2.7:

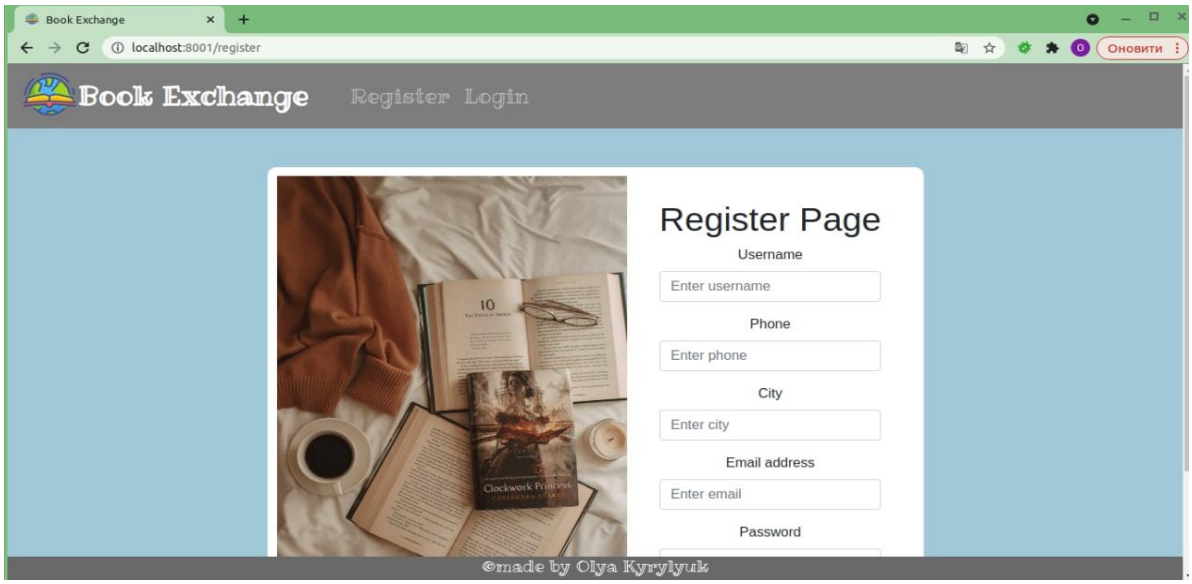


Рисунок 2.7 – Сторінка реєстрації в веб-системі

На рисунку 2.8 зображено першу сторінку після авторизації або реєстрації акаунту:

Рисунок 2.8 – Сторінка перегляду власних книг

Меню сторінки для авторизованого користувача складається з таких полів як:

– Mybooks – сторінка перегляду та додання власних книг;

– BookWorld – сторінка перегляду книг інших користувачів;

– Notifications – сторінка перегляду запитів на обмін книгами;

– Logout – вихід з системи.

Також користувач бачить банерну картинку та можливість додати нову книгу при натисненні на кнопку.

На рисунку 2.9 зображено другу частину сторінки перегляду власних книг:

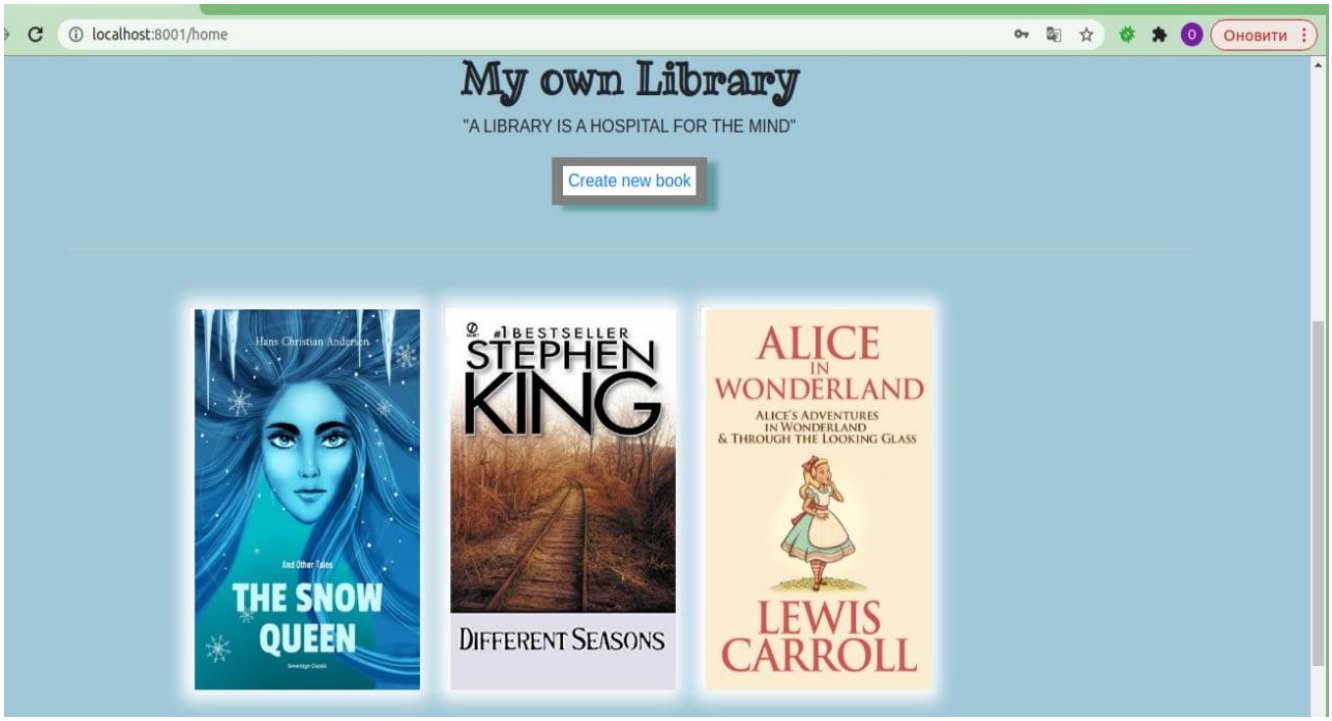


Рисунок 2.9 – Друга частина сторінки перегляду власних книг

При натисненні на картинку, користувач має змогу перейти на деталі картинки, та вирішити створювати обмін книгами, чи ні. Далі в залежності від вибору другого користувача (користувач може як прийняти, так і відхилити запит на обмін) буде відбуватися обмін книгами.

Для можливості обміну користувач повинен переглядати книги інших користувачів.

На рисунку 2.10 зображено сторінку перегляду всіх книг для пошуку книги для обміну:

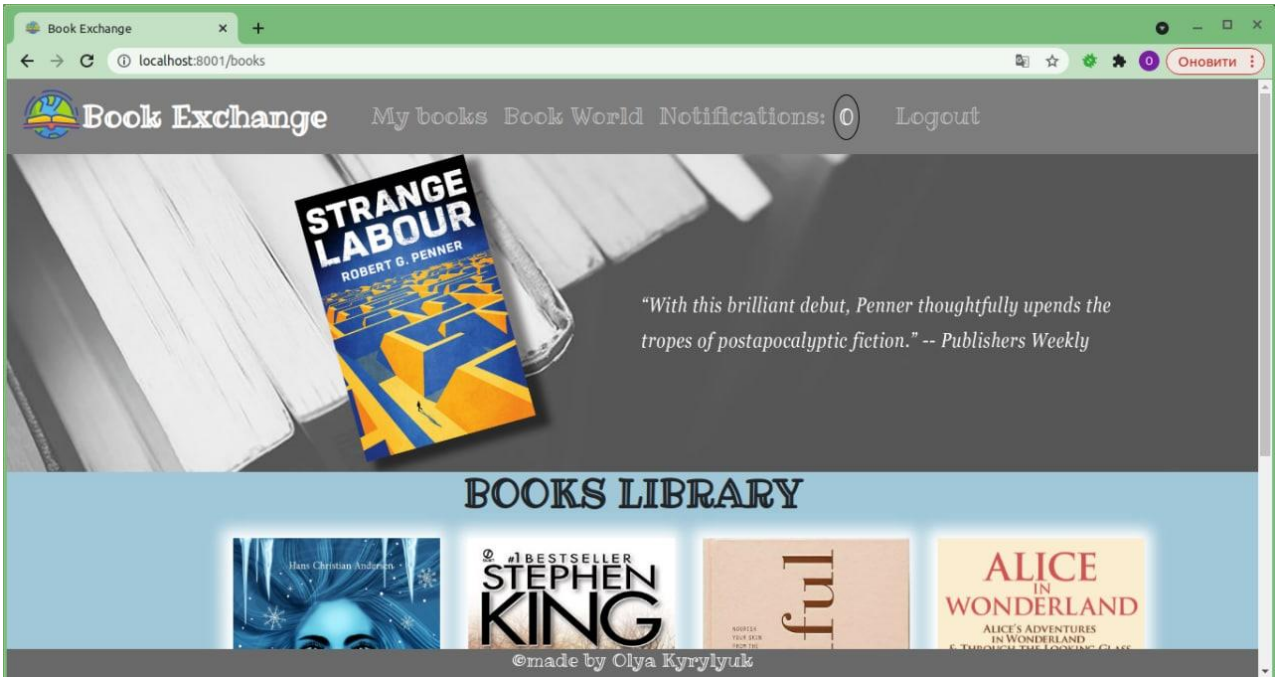


Рисунок 2.10 – Сторінка перегляду всіх книг для обміну

На сторінці 2.11 зображено сторінку додання книги:

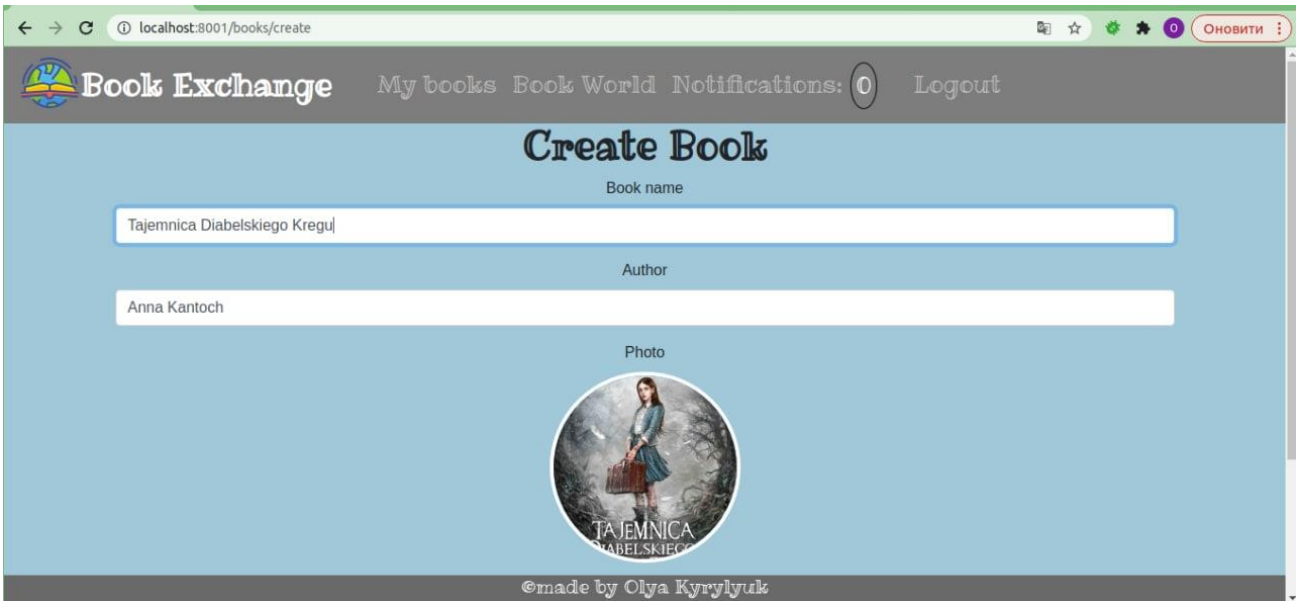


Рисунок 2.11 – Сторінка додання книги

При натисненні мишкою на зображення книги можна перейти на деталі книги, які можна переглянути, а саме:

- автор;
- назва книги;
- зображення.

Сторінка перегляду деталей книги зображено на рисунку 2.11 нижче:

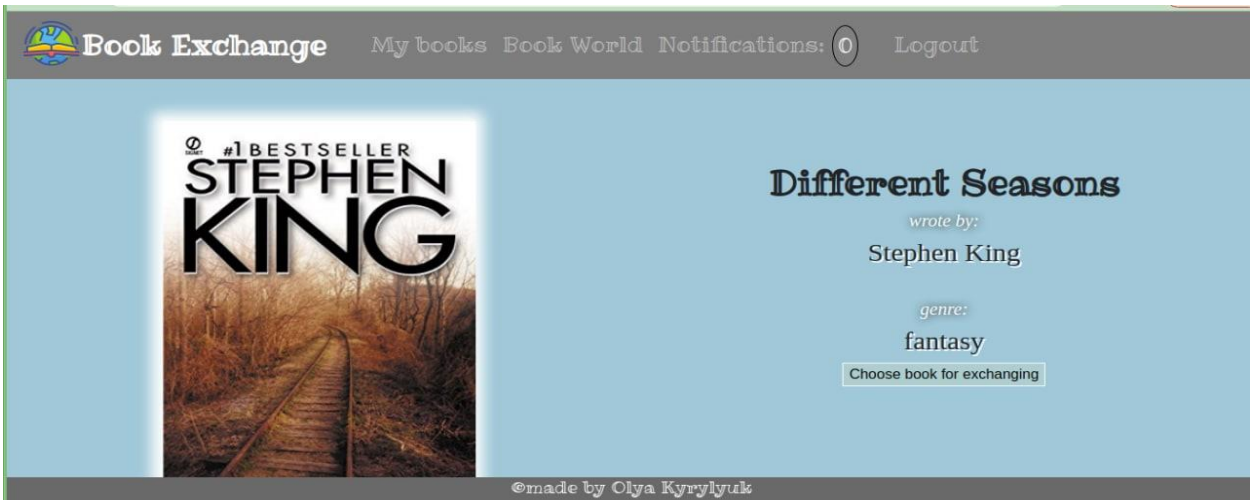


Рисунок 2.11 – Сторінка перегляду деталей книги

При натисненні на кнопку обміну книгою, з’являється повідомлення про успішне відправлення запиту на обмін книгами. Повідомлення зображено на рисунку 2.12:

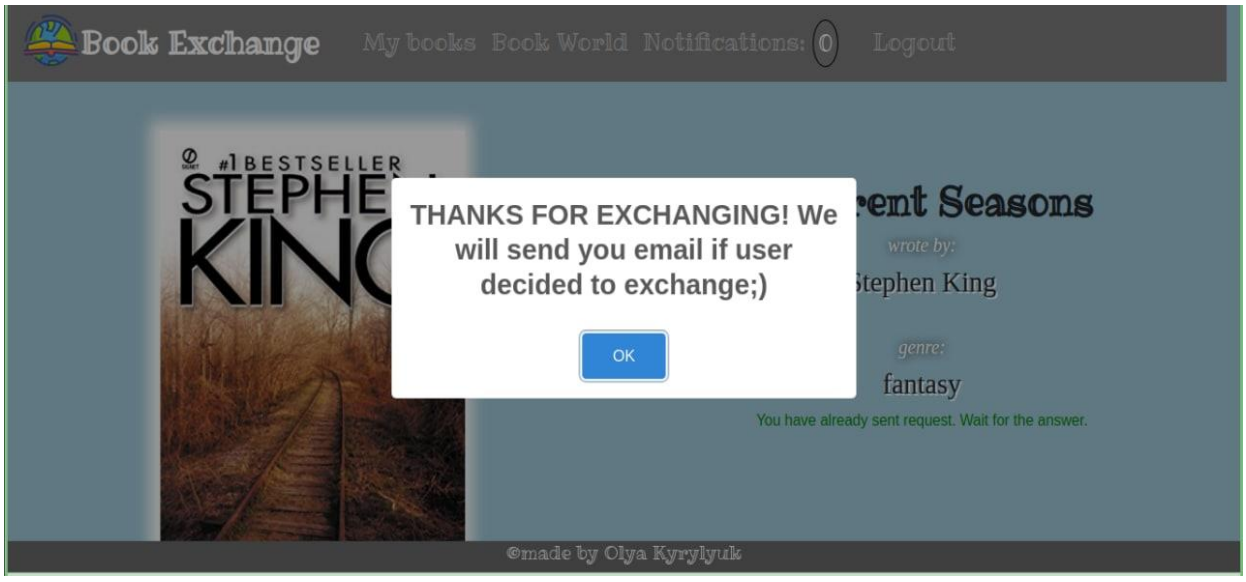


Рисунок 2.12 – Повідомлення про успішний запит на обмін книгами

Після натиснення на кнопку в повідомленні користувач бачить замість кнопки повідомлення про відправлений запит та прохання зачекати, поки власник книги не погодиться на обмін книги.

Сторінка перегляду повідомлення зображено на рисунку 2.13:

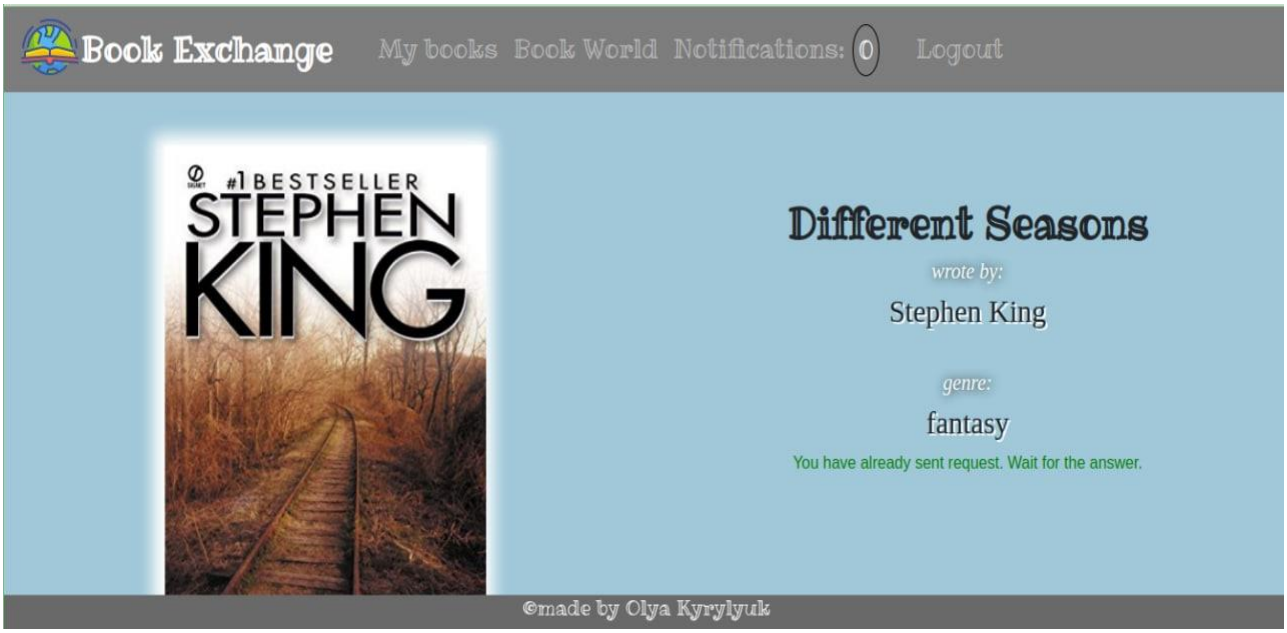


Рисунок 2.13 – Сторінка перегляду книги з повідомленням

Сторінка перегляду всіх запитів на обмін книгами зображено на рисунку 2.14:

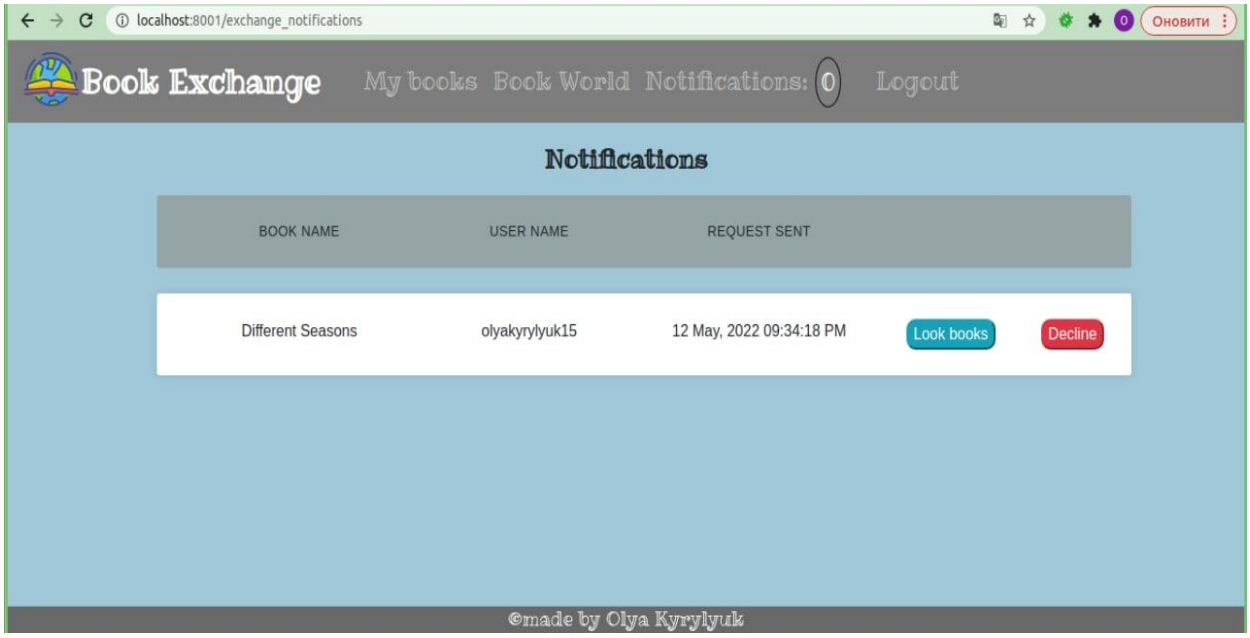


Рисунок 2.14 – Сторінка перегляду всіх запитів на обмін книгами

На рисунку 2.15 зображено книги, які користувач може вибрати для обміну.

Рисунок 2.15 – Сторінка перегляду книг іншого користувача

Після прийняття запиту користувачам на gmail приходять повідомлення про успішний обмін книгами. Результати отримання повідомлення на пошту зображено на рисунку 2.16 нижче:

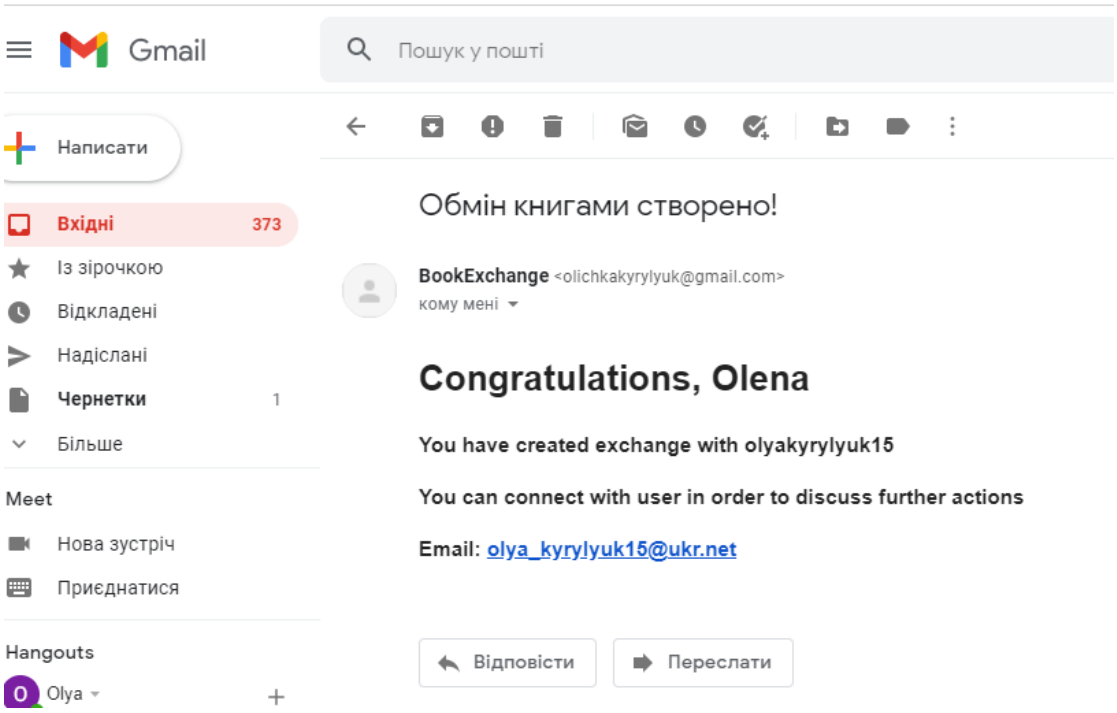


Рисунок 2.16 – Повідомлення про успішний обмін книгами

2.5 Аналіз та вибір технологій і методів реалізації системи

Для розробки веб-додатку було обрано мову програмування – PHP.

PHP є найпопулярнішою мовою сценаріїв на стороні сервера. Він розроблений для веб-розробки та програмування загального призначення в 1994 році Расмусом Лердорфом. За два десятиліття розвитку PHP пережив як злети, так і падіння. Зараз PHP управляється The PHP Group і знаходиться в безперервному розвитку. PHP означає Hypertext Preprocessor, яка була змінена з початкової назви «Персональна домашня сторінка».

PHP в основному використовується в поєднанні з HTML-кодом, системою керування веб-контентом, системами веб-шаблонів та іншими популярними веб-фреймворками.

Мова PHP обробляється сервером або загальним інтерфейсом шлюзу (CGI). У будь-якому випадку, його можна використовувати для створення веб-додатка, де PHP-код завжди виконується на стороні сервера. Також, можна виконувати PHP-код за допомогою інтерфейсу командного рядка (CLI). Крім того, існує можливість використовувати його для реалізації графічного додатка, який є автономним за своєю природою.

Існує безліч фремворків для розробки веб-сайту на мові PHP. Для дипломного проекту було обрано PHPфреймворк – Laravel.

Laravel – це PHP-фреймворк з відкритим вихідним кодом, який є надійним і легким для розуміння. Він відповідає шаблону проектування модель-представлення-контролер. Laravel повторно використовує існуючі компоненти різних фреймворків, що допомагає у створенні веб-додатків. Таким чином, розроблений веб-додаток є більш структурованим і прагматичним.

Laravel пропонує багатий набір функціональних можливостей, який включає основні функції PHP-фреймворків, таких як CodeIgniter, Yii та інших мов програмування, як-от Ruby on Rails. Laravel має дуже багатий набір функцій, які підвищать швидкість веб-розробки.

Переваги Laravel.

Laravel пропонує наступні переваги, при створенні веб-додатку на його основі цього:

– веб-додаток стає більш масштабованим завдяки PHP – фреймворку Laravel;

– при розробці веб-додатка економиться значний час, оскільки Laravel повторно використовує компоненти з іншого фреймворку при розробці веб-додатка;

– він включає простори імен та інтерфейси, таким чином допомагає організувати ресурси та керувати ними.

Інструменти, які є перевагами фреймфорку Laravel:

1. Composer.

Composer – це інструмент, який включає в себе всі залежності та бібліотеки. Це дозволяє користувачеві створювати проект щодо згаданого фреймворку (наприклад, тих, що використовуються під час встановлення Laravel). Сторонні бібліотеки можна легко встановити за допомогою composer.

Усі залежності зазначаються у файлі composer.json, який розміщений у вихідній папці.

2. Artisan.

Інтерфейс командного рядка, який використовується в Laravel, називається Artisan. Він включає в себе набір команд, які допомагають створити веб-програму. Ці команди включені з фреймворку Symphony, що призводить до появи додаткових функцій у Laravel 5.1.

3. Особливості Laravel.

Laravel пропонує наступні ключові функції, що робить його ідеальним вибором для розробки веб-програм

4. Модульність.

Laravel надає 20 вбудованих бібліотек і модулів, які допомагають покращити програму.

Кожен модуль інтегровано з менеджером залежностей Composer, що полегшує оновлення.

5. Перевірка.

Laravel включає функції та помічники, які допомагають у тестуванні за допомогою різних тестових випадків. Ця функція допомагає підтримувати код відповідно до вимог.

6. Маршрутизація.

Laravel забезпечує гнучкий підхід до користувача для визначення маршрутів у веб-додатку. Маршрутизація допомагає краще масштабувати додаток і підвищує його продуктивність.

7. Управління конфігурацією.

Веб-додаток, розроблений на Laravel, працюватиме в різних середовищах, а це означає, що його конфігурація буде постійно змінюватися. Laravel забезпечує послідовний підхід до ефективної обробки конфігурації.

8. Query Builder та ORM.

Laravel включає в себе конструктор запитів, який допомагає виконувати запити до баз даних за допомогою різних простих методів ланцюга. Він забезпечує реалізацію ORM (Object Relational Mapper) і ActiveRecord під назвою Eloquent.

9. Schema Builder.

Schema Builder підтримує визначення та схему бази даних у коді PHP. Він також відстежує зміни щодо міграцій бази даних.

10.Template Engine.

Laravel використовує механізм Blade Template, легку мову шаблонів, що використовується для розробки ієрархічних блоків і макетів із попередньо визначеними блоками, які містять динамічний вміст.

11.E-mail.

Laravel включає в себе клас пошти, який допомагає надсилати пошту з багатим вмістом і вкладеннями з веб-програми.

12.Authentication.

Аутентифікація користувача є поширеною функцією у веб-програмах. Laravel полегшує розробку аутентифікації, оскільки включає такі функції, як реєстрація, забутий пароль та надсилання нагадувань про пароль.

13.Redis.

Laravel використовує Redis для підключення до існуючого сеансу та кешу загального призначення. Redis безпосередньо взаємодіє із сесією.

14.Queues.

Laravel включає служби черги, як-от надсилання електронної пошти великій кількості користувачів або вказане завдання Cron. Ці черги допомагають виконувати завдання легше, не чекаючи завершення попереднього завдання.

15.Event and Command Bus.

Laravel включає eventandcommandbus, яка допомагає виконувати команди та відправляти події простим способом. Команди в Laravel діють відповідно до життєвого циклу програми.

PhpMyAdmin – це безкоштовний інструмент, написаний на PHP, який надає інтерфейс для роботи з базами даних MySQL. За допомогою цієї програми можна створювати, змінювати та видаляти записи, а також імпортувати та експортувати таблиці з бази даних MySQL. Можливо також виконувати запити MySQL, оптимізувати та відновити базу даних та багато інших завдань.

Функції PhpMyAdmin:

а) інтуїтивно зрозумілий веб-інтерфейс;

б) підтримка більшості функцій MySQL:

1) переглядати та видалядати бази даних, таблиці, представлення даних, поля та індекси;

2) створювати, копіювати, скидати, перейменовувати та змінювати бази даних, таблиці, поля та індекси;

3) обслуговувати сервера, бази даних і таблиці з пропозиціями щодо конфігурації сервера;

4) виконувати, редагувати та створювати закладки для будь-яких операторів SQL, навіть пакетних запитів;

5) керуватиобліковими записами та привілеями користувачів MySQL;

6) керувати збереженими процедурами та тригерами;

7) імпортувати дані із CSV та SQL;

8) експортувати дані у різні формати: CSV, SQL, XML, PDF та інші;

9) адмініструвати кількох серверів;

10) створювати графіки макета вашої бази даних у різних форматах;

11) створювати складних запитів за допомогою Query-by-example (QBE);

12) глобальношукати у базі даних або її підмножині;

13) перетворювати збережені даних у будь-який формат за допомогою набору попередньо визначених функцій.

У даному розділі було проведено аналіз та вибір технологій і методів реалізації системи. Також, було спроектовано інтерфейс користувача та база даних, які буду необхідні при розробці програмного продцукту.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Для того, щоб зв'язати базу даних з проектом, необхідно в файлі .env прописати налаштування, подані на рисунку 3.1:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dbBookExchange
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 3.1 – Налаштування бази даних в проекті

Також в проекті в розділі «Database» необхідно додати налаштування, зображені на рисунку 3.2:

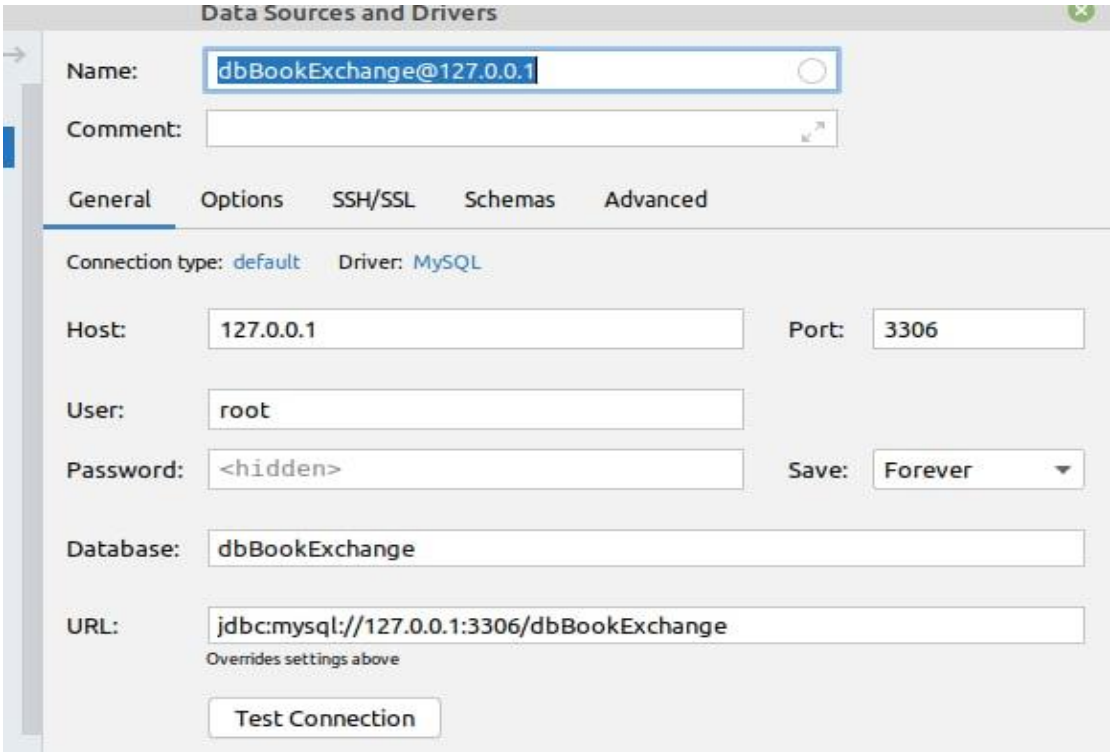


Рисунок 3.2 – Налаштування бази в розділі Database

Після створення міграцій, база даних в проекті виглядатиме відповідно до зображення 3.3:

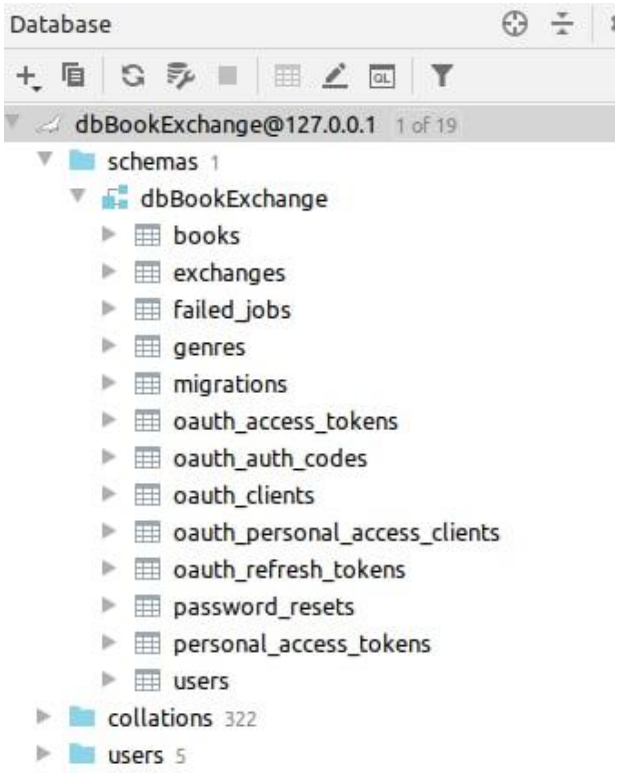


Рисунок 3.3 – Розділ Databaseпісля налаштувань

На рисунку 3.4 зображений OpenServer.

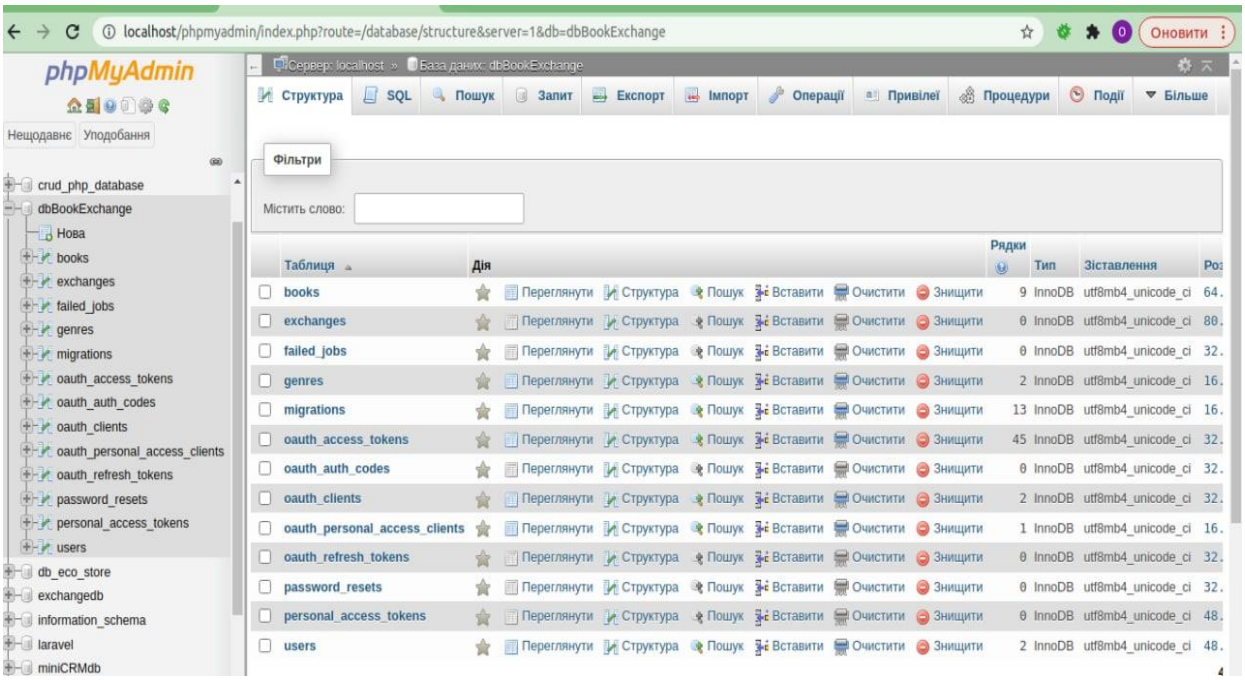


Рисунок 3.4 - База даних в OpenServer

Тепер дані можна зручно відслідковувати після зміни інформації в системі.

3.2 Розробка програмних модулів

Структура проекту, реалізованого для отримання дипломного програмного продукту зображена на рисунку 3.5:

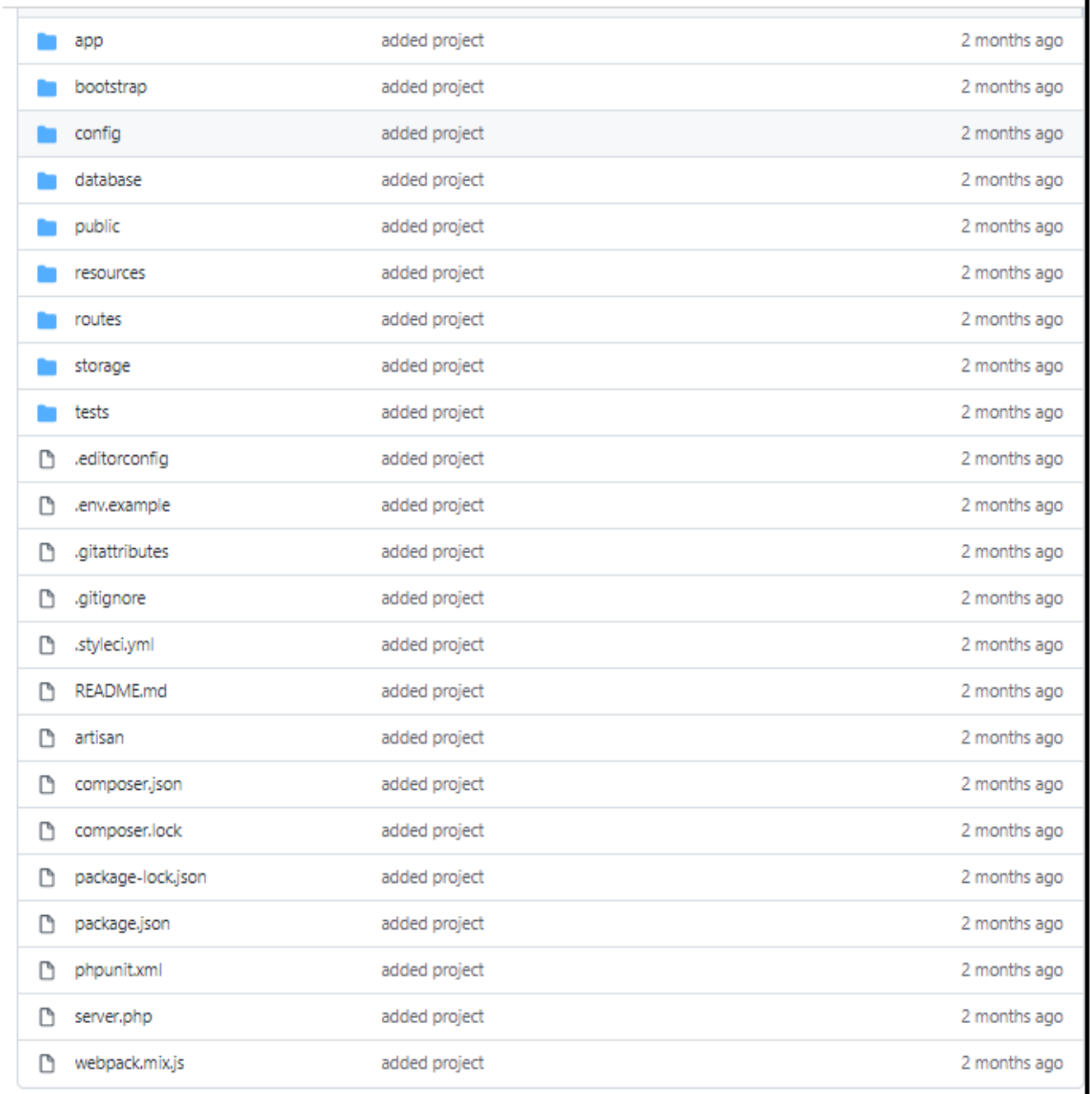


Рисунок 3.5 – Структура проекту

Для побудови таблиць бази даних створена папка «database», в якій можна додавати дані, що стосуються бази даних.

Нижче нарисунку 3.6 зображено папку «database»:

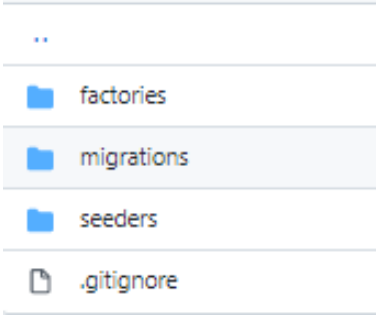


Рисунок 3.6 – Структура папки папку «database»

– migrations – папка для створення самих таблиць та полів в таблиці;

– seeders – папка для створення фейкових даних для тестування в необхідній кількості;

– factories – папка для створення фейкових даних, яка взаємопов’язана з папкою seeders.

Створення міграції на прикладі таблиці Books:

```
<?php
        use Illuminate\Database\Migrations\Migration;
        use Illuminate\Database\Schema\Blueprint;
        use Illuminate\Support\Facades\Schema;
        class CreateBooksTable extends Migration
        {
            /**
             * Run the migrations.
             *
             * @return void
             */
            public function up()
            {
                Schema::create('books', function (Blueprint $table) {
        $table->id();
                    $table->string('name')->required();
                    $table->string('slug')->unique();
                    $table->string('author')->required();
                    $table->string('photo')->nullable();
                    $table->bigInteger('user_id')->unsigned()->nullable();
                    $table->foreign('user_id')->references('id')->on('users')-
        >onDelete('set null');
                    $table->bigInteger('genre_id')->unsigned()->nullable();
                    $table->foreign('genre_id')->references('id')->on('genres')-
        >onDelete('set null');
        $table->timestamps();
                });
            }
```

```
        /**
         * Reverse the migrations.
         *
         * @return void
         */
        public function down()
        {
            Schema::dropIfExists('books');
        }
    }
```

В папці routes можна прописувати шляхи. Восновномудля розробки веб-сайту використовується файл «web.php».

Структура папки зображена нижче на рисунку 3.7:

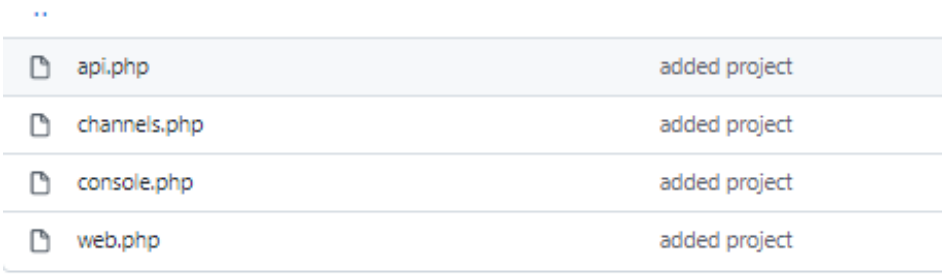


Рисунок 3.7 - Структура папки папку «web.php»

В папці «App», що на рисунку 3.8 знаходиться основна логіка веб-додатку. За технологією MVCмоделі знаходяться в папці Models:

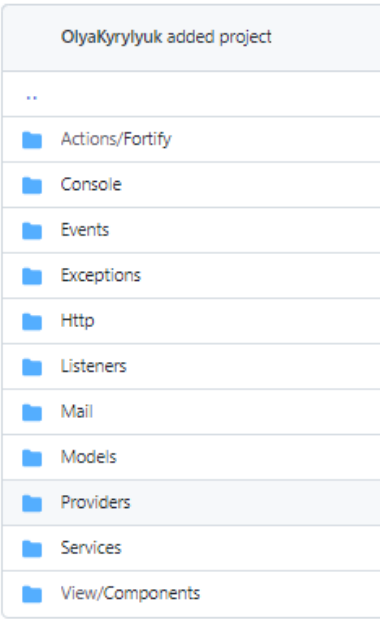


Рисунок 3.8 - Структура папки папку «App»

Приклад моделі:

```
<?php
```

```
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Book extends Model
{
    use HasFactory;
protected $fillable = [
        'name', 'photo','slug','author', 'user_id', 'genre_id'
];
    public function user()
    {
        return $this->belongsTo(User::class);
}
    public function genre()
    {
        return $this->belongsTo(Genre::class);
}
    public function exchanges()
    {
        return $this->hasMany(Exchange::class);
}
}
```

Контролери знаходяться в папці HTTP, яка зображена на рисунку 3.9:

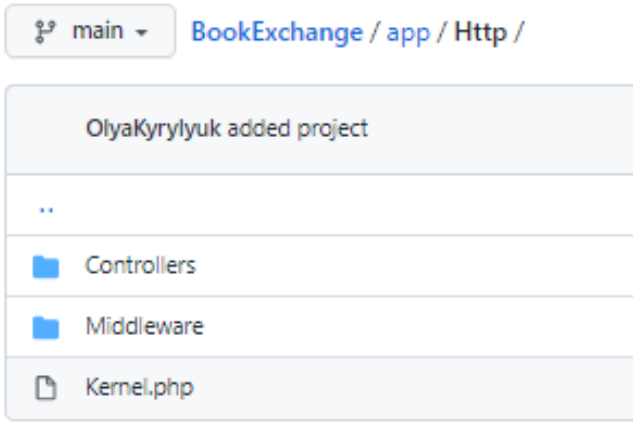


Рисунок 3.9 - Структура папки папку «HTTP»

Приклад створення контролера:

```
<?php
        namespace App\Http\Controllers;
        use App\Models\Genre;
        use App\Models\Book;
        use App\Models\Exchange;
        use App\Models\User;
        use App\Services\BookService;
        use Illuminate\Http\Request;
        use App\Events\SuccessExchange;
        class BookController extends Controller
        {
            private $bookService;
            public function __construct(BookService $bookService)
        {
                $this->bookService = $bookService;
```

```
    }
    public function home()
    {
        $user = auth()->user();
        $books = Book::with(['genre', 'user'])->where('user_id', $user-
>id)->orderBy('created_at', 'desc')->get();
        return view('home', ['books'=>$books]);
}
public function index(){
        $books = Book::with(['genre', 'user'])->orderBy('created_at',
'desc')->get();
        return view('books.index', ['books'=>$books]);
}
    public function show($slug)
    {
       $data =  $this->bookService->showOneBook($slug);
$book = $data[0];
        $current_user = $data[1];
        $exchange = $data[2];
        return view('/books/single_book',['book'=>$book,
'current_user'=>$current_user, 'exchange'=> $exchange]);
}
public function create(){
        $genres = Genre::all();
        return view('books.create', ['genres'=>$genres]);
}
    public function store(Request $request)
{
        $input = $request->except(['genres']);
        $this->bookService->saveBook($input, $request->genres, $request-
>hasFile('photo'), $request->name, $request->photo);
        return redirect()->route('books.index');
}
    public function bookRequest($slug)
    {
        $this->bookService->sendRequest($slug);
return redirect()->back()->with('message', 'IT WORKS!');
}
    public function choose_exchange_book($id)
{
        $exchange = Exchange::where('id', $id)->first();
        $user_profile = User::with('books')->where('id', $exchange-
>user1_id)->get();
        return view('books.choose_exchange_book',
['user_profile'=>$user_profile, 'exchange'=>$exchange]);
}
    public function accept_exchange_book($id, $book_id)
{
        $exchange = Exchange::where('id', $id)->first();
        $exchange->book2_id = $book_id;
$exchange->status = 'success';
        $exchange->save();
        $user_from = User::where('id',$exchange->user1_id)->first();
        $book1 = Book::where('id', $exchange->book1_id);
        $book2 = Book::where('id', $exchange->book2_id);
        event(new SuccessExchange(auth()->user(), $user_from, $book1,
$book2));
return redirect('/home')->with('message', 'Request was sent to user! Wait
for the message on your email about applying or declining request!');
}
    public function welcome()
    {
        $books = Book::with(['genre', 'user'])->orderBy('created_at',
```

```
        'desc')->get();
                return view('books.welcome', ['books'=>$books]);
        }
        }
```

Для того, щоб не навантажувати контролер, великі частини коду виносять в Services, які знаходяться в папці «App», що на рисунку 3.10:

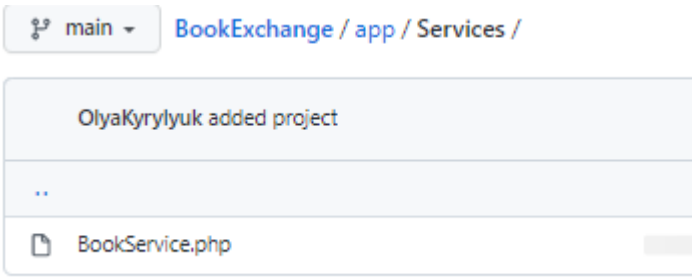


Рисунок 3.10 - Структура папки «Services»

Приклад створення Сервісу:

```
<?p
hp
        namespace App\Services;
        use App\Models\Book;
        use App\Models\Exchange;
        use Illuminate\Support\Facades\Auth;
        use Illuminate\Support\Str;
        class BookService
        {
            public function saveBook($input, $genres, bool $hasFile, $name,
        $photo)
        {
                $input['user_id'] = Auth::user()->id;
        $input['genre_id'] = $genres;
                $input['slug'] = Str::slug($name, '-');
        $newPhotoName = "";
        if($hasFile) {
                    $newPhotoName = time() . '-' . $name ."." . $photo-
        >extension();
                    $photo->move(public_path('images/books'), $newPhotoName);
        }
                $input['photo'] =  $newPhotoName;
                Book::create($input);
            }
            public function sendRequest($slug)
            {
                $current_user = $user = auth()->user();
                $book_request = Book::with('user')->where('slug', $slug)->get();
        Exchange::create([
                    'user1_id' => $current_user->id,
                    'user2_id' => $book_request[0]->user->id,
                    'book1_id' => $book_request[0]->id,
        ]);
            }
            public function showOneBook($slug)
```

```
    {
        $current_user = $user = auth()->user();
        $book = Book::with('user')->where('slug',$slug)->first();

$exchange = Exchange::with(
            ['book' => function($q) use ($book){
                $q->where('id', $book->id);
}
            ],

            ['user' => function($q) use ($current_user){
                $q->where('id', $current_user->id );
}
            ])->where('user1_id', $current_user->id)
->get();

//        $exchange = Book::with(['exchanges', 'user'=>function($q) use
($current_user){
//                $q->where('id', $current_user->id );
//            }])->get();

if($hasFile) {
            $newPhotoName = time() . '-' . $name ."." . $photo-
>extension();
            $photo->move(public_path('images/books'), $newPhotoName);
}
        $input['photo'] =  $newPhotoName;
        $current_user = $user = auth()->user();
        $book = Book::with('user')->where('slug',$slug)->first();
}
```

Всередині цієї папки можна побачити папки, призначені для відображення сайту користувачеві, переглянути можна на рисунку 3.11, що зображений нижче:

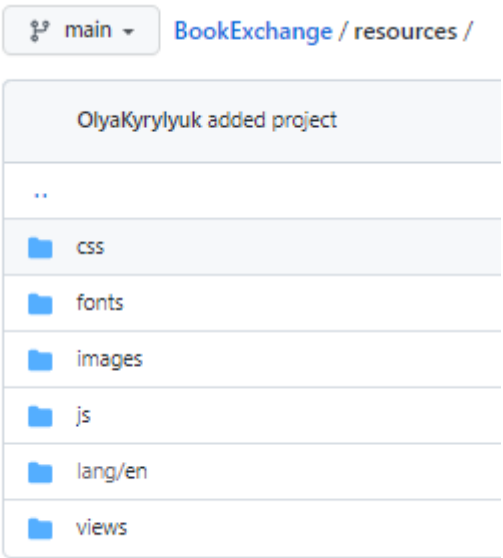


Рисунок 3.11 - Структура папки «recources»

В папці «views»прописується код для створення HTML-сторінок. Приклад сторінки, яка є основним шаблоном для його використання в інших сторінках:

```
<!doctyp
e html>
            <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
            <head>

            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <!-- CSRF Token -->
            <meta name="csrf-token" content="{{ csrf_token() }}">
            <title>Book Exchange</title>

            <!-- Styles -->
            <link rel="icon" href="https://img.icons8.com/external-wanicon-
            lineal-color-wanicon/64/000000/external-book-library-wanicon-lineal-
            color-wanicon-3.png">

            <link href="{{ URL::asset('css/sweetalert.css') }}" rel="stylesheet"
            type="text/css"/>
            <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
            js/2.4.1/core.js"></script>
            <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
            js/2.4.1/core.js"></script>

            <link rel="stylesheet"
            href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.m
            in.css" integrity="sha384-
            Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
            crossorigin="anonymous">

            <link href="{{ asset('css/app.css') }}" rel="stylesheet">
            </head>
            <body>
            @include('menu')
            <div class="content">
                    @yield('content')
            </div>
            <div class="footer">

            <div class="fixed-bottom">
            <div class="footer_center">
            <h5>&copy;made by Olya Kyrylyuk</h5>
            </div>
            </div>
            </div>

            <!-- Scripts -->
            <script type="text/javascript"
            src="{{asset('js/jQuery.js')}}"></script>
            <script src="{{ asset('js/app.js') }}"></script>
            @yield('JS')

            </body>
            </html>
```

Стилі для сторінок додаються в папці «css». Зображена папка на рисунку 3.12:

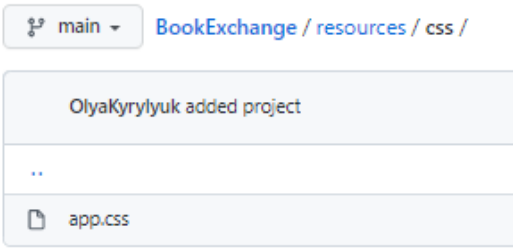


Рисунок 3.12 - Структура папки «css»

Приклад додання стилів для веб-сторінок:

```
@font-
face {
            font-family: 'RibeyeMarrow';
            src: url('/fonts/RibeyeMarrow-Regular.woff') format('woff');
        font-weight: normal;
            font-style: normal;
        }
        html, body {
            text-align: center;
            background-color: #a0c8d9;
            height: 100%;
        }
        .navText {
            font-family: RibeyeMarrow;
            font-weight: 700;
            font-style: normal;
            font-size: 33px;
            color: #fff ;
            margin-right: 40px;
        }
        .navText:hover{
            text-decoration: none;
            color: #fff;
        }
        .navRight a {
            padding: 5px;
            font-family: RibeyeMarrow;
            font-weight: 500;
            font-style: normal;
            font-size: 25px;
            color: #fff;
            text-decoration: none;
        }
        .imgLogin {
            width: 400px;
            height: 600px;
        }
            font-family: RibeyeMarrow;
            font-weight: 500;
            font-style: normal;
            font-size: 25px;
            color: #a0c8d9 !important;
```

```
        text-decoration: none;
    }
    .white_block {
        padding: 10px;
        background-color: #fff;
        display: flex;
        flex-direction: "row";
        justify-content: center;
        max-width: 750px;
        margin: 0 auto;
        border: 1px solid whitesmoke;
        border-radius: 10px;
    }
    .imgRegiserLogin {
        width: 300px;
        height: 400px;
    }
    .imgLogin {
        width: 400px;
        height: 600px;
    }
```

3.3 Керівництво користувача

При вході на сайт користувач бачить гостьову сторінку, яка є обмеженою, в порівнянні з сторінками для авторизованих користувачів, сторінкуможна переглянути нижче на рисунку 3.13:

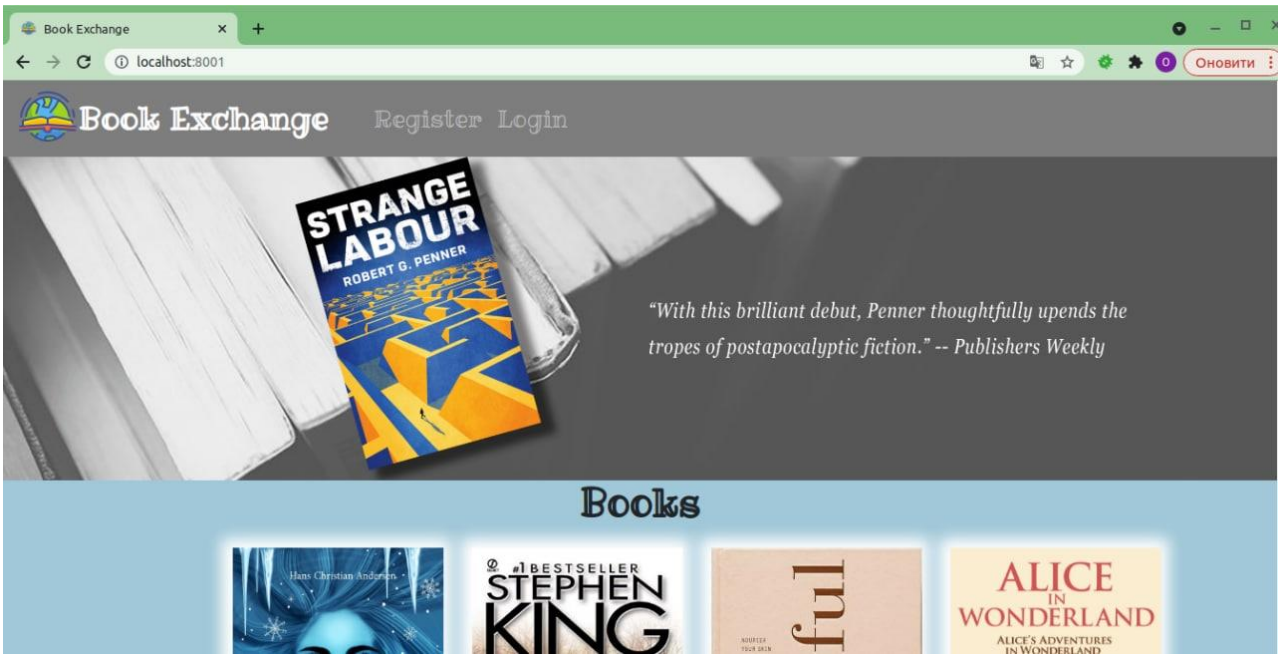


Рисунок 3.13 – Гостьова сторінка

Меню неавторизованого користувача зображено на рисунку 3.14:

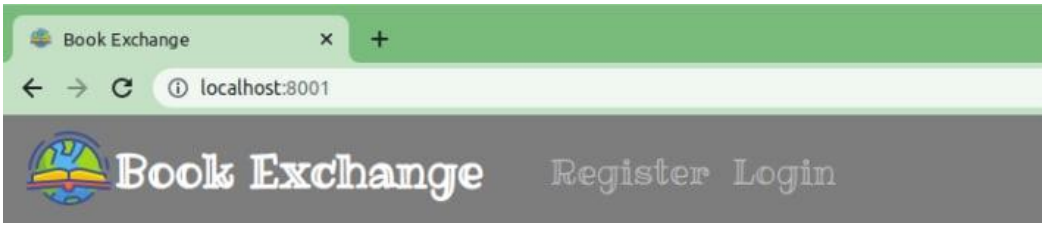


Рисунок 3.14 – Меню неавторизованого користувача

На меню є дві вкладки реєстрації та авторизації. При натисненні на реєстрацію відкриється сторінка реєстрації з полями:

- username;
- phone;
- city;
- email address;
- password.

Для реєстрації вводяться поля, які потрібно заповнити для того, щоб користуватися системою в повному обсязі. В разі помилкового введення даних користувач не зможе зареєструватися.

Сторінка реєстрації зображена на рисунку 3.15:

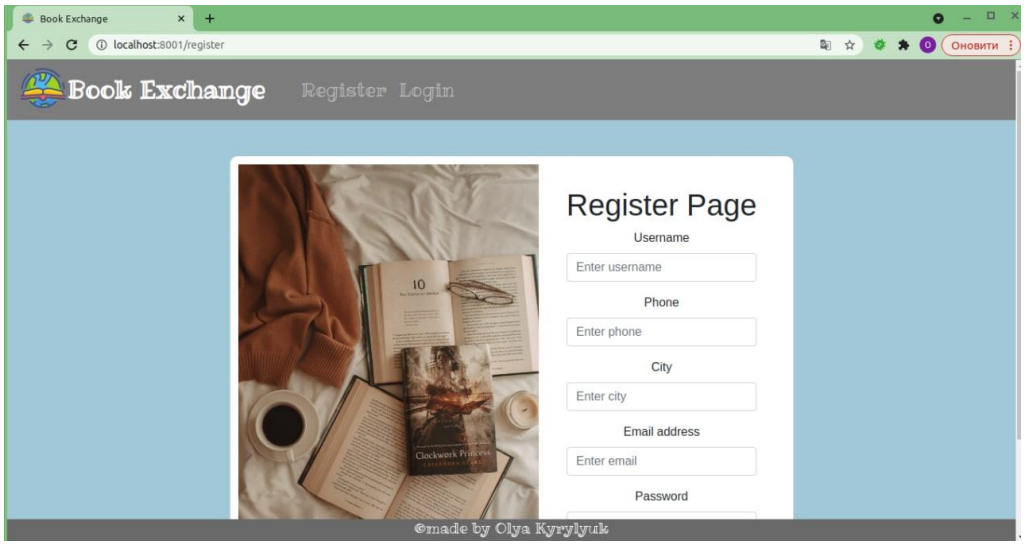


Рисунок 3.15 – Сторінка реєстрації

Якщо користувач був попередньо зареєстрований, необхідно лише авторизуватися.

Поля, які необхідно заповнити для авторизації: emailaddress та password.

Також сторінка авторизації містить посилання на сторінку реєстрації, якщо користувач випадково зайшов надану сторінку. При натисненні на кнопку система перевіряє чи існує даний користувач в системі, і якщо email та пароль співпадають, читач може перейти до використання сайту в повному обсязі.

Для авторизації потрібно в меню натиснути на потрібний пункт.

При натисненні відкриється сторінка авторизації, що зображена на рисунку 3.16:

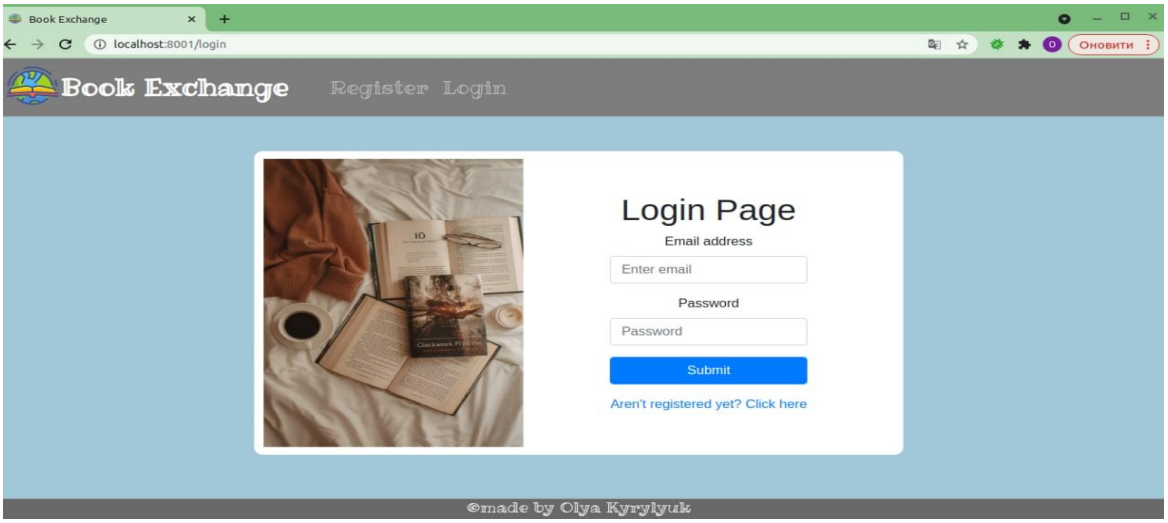


Рисунок 3.16 – Сторінка авторизації

На даній сторінці можна перейти на сторінку реєстрації при необхідності, або ж ввести свої дані та перейти в систему.

Після авторизації меню авторизованого користувача зображено на рисунку 3.17:

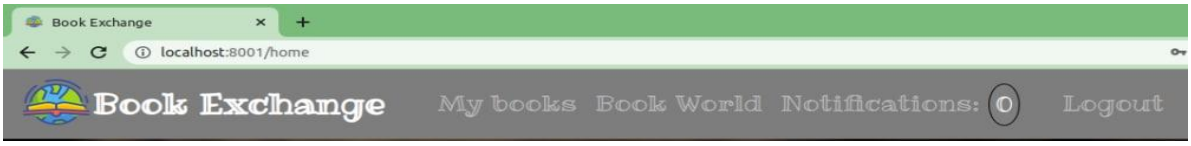


Рисунок 3.17 – Меню авторизованого користувача

Список меню складається з:

– MyBooks – сторінка перегляду та додання власних книг;

– BookWorld – сторінка перегляду всіх існуючих книг в системі;

– Notifications – перегляд запитів стосовно обміну книгами. Також цифра означає кількість непрочитаних запитів. При переході на сторінку – цифра обнуляється;

– Logout –вихід з системи.

Перша, сторінка, яка відображається на сторінці – сторінка власних книг, що зображена на рисунку 3.18:

Рисунок 3.18 – Перша частина домашньої сторінки власних книг

Друга частина сторінкизображена на рисунку 3.19:

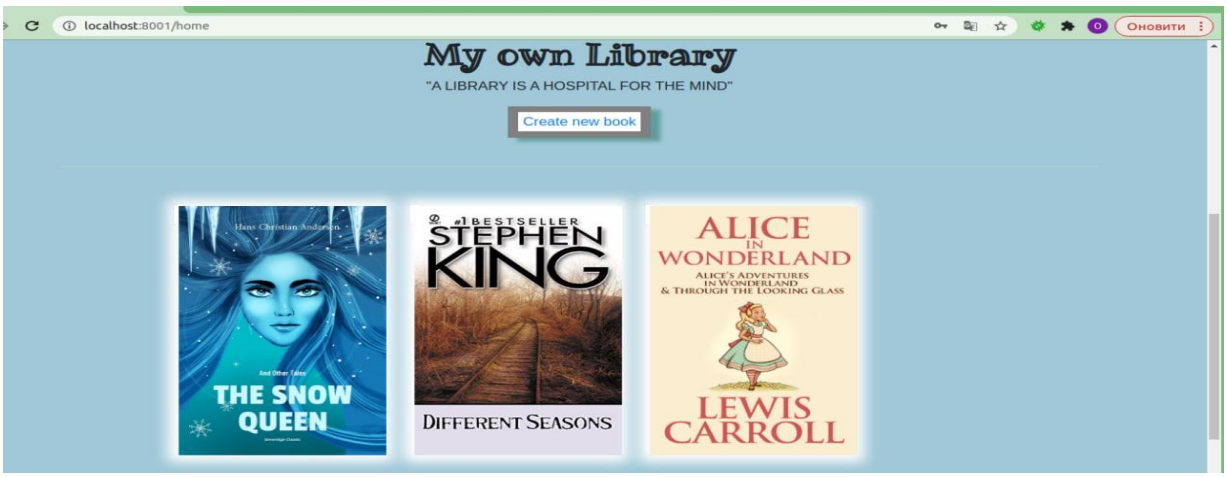


Рисунок 3.19 – Друга частина домашньої сторінки

Кожний користувач може додавати власні книги на цій сторінці, а також переглядати кожну сторінку з детальною інформацією.

Сторінка створення власної книги на рисунку 3.20:

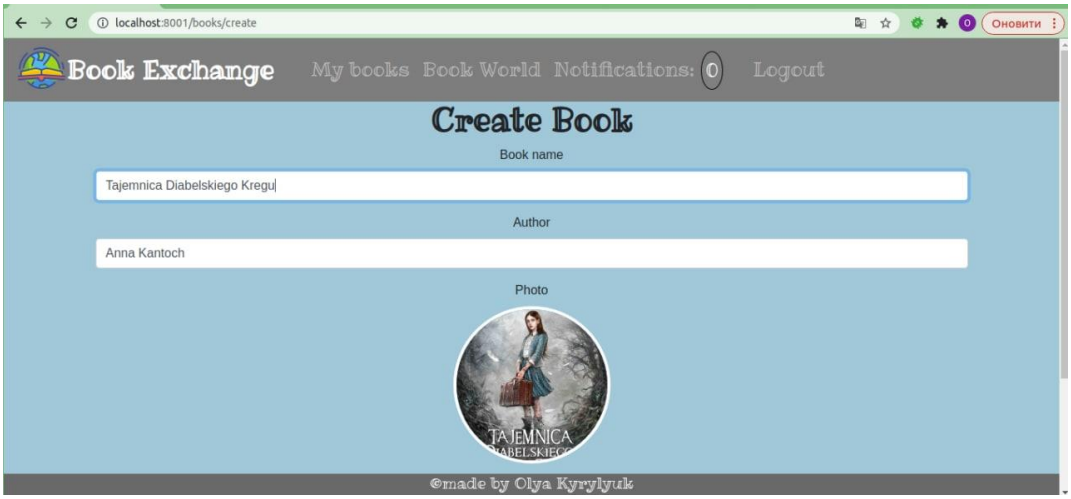


Рисунок 3.20 – Сторінка додання власної книги для обміну

Потрібно заповнити такі поля:

– назва книги;

– автор;

– картинка книги;

– жанр книги, який обирається з переліку доступних жанрів.

На сторінці з всіма книгами, що наведена на рисунку 3.21 можна обирати книгу для обміну:

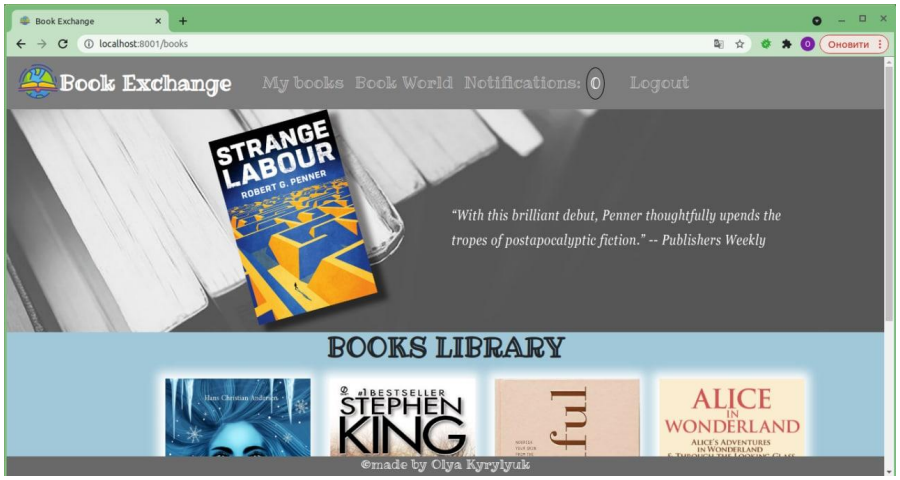


Рисунок 3.21 – Сторінка з всіма книгами

При натисненні на картинку в списку книг, відкривається сторінка з детальною інформацією, дану сторінку зображено на рисунку 3.22:

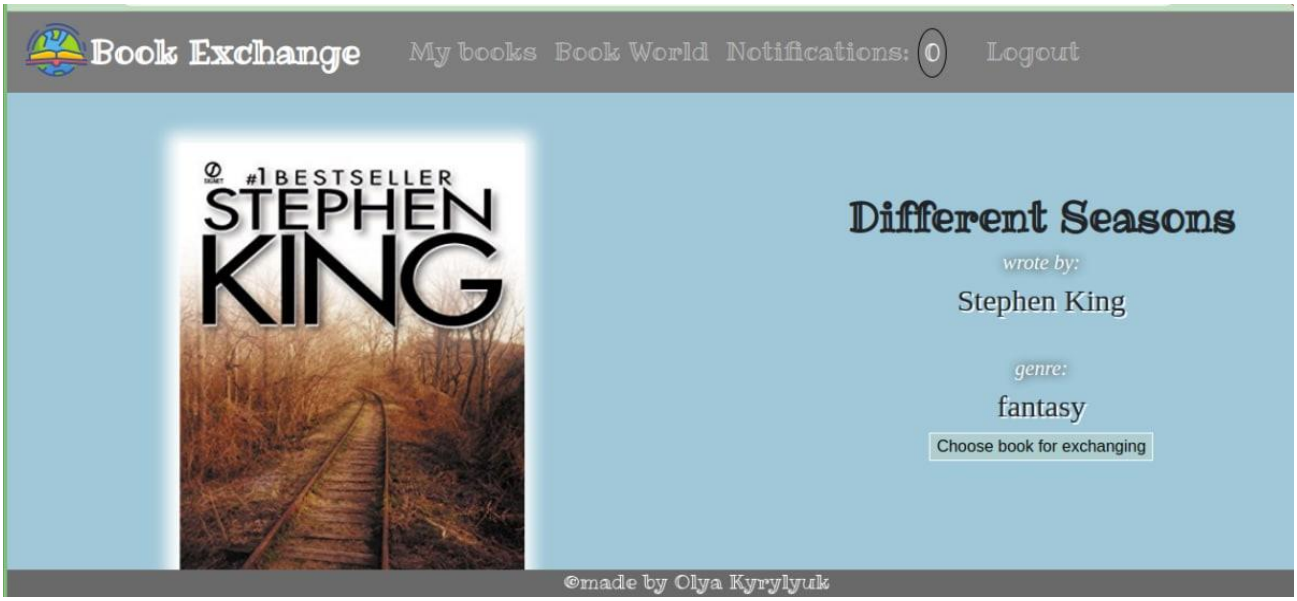


Рисунок 3.22 – Сторінка з детальною інформацією про книгу

Для того, щоб створити запит на обмін книгою, необхідно натиснути на кнопку «Choosebook for exchanging», що зображена на сторінці. Після того виведеться повідомлення про успішне відправлення запиту, яке зображено на рисунку 3.23:

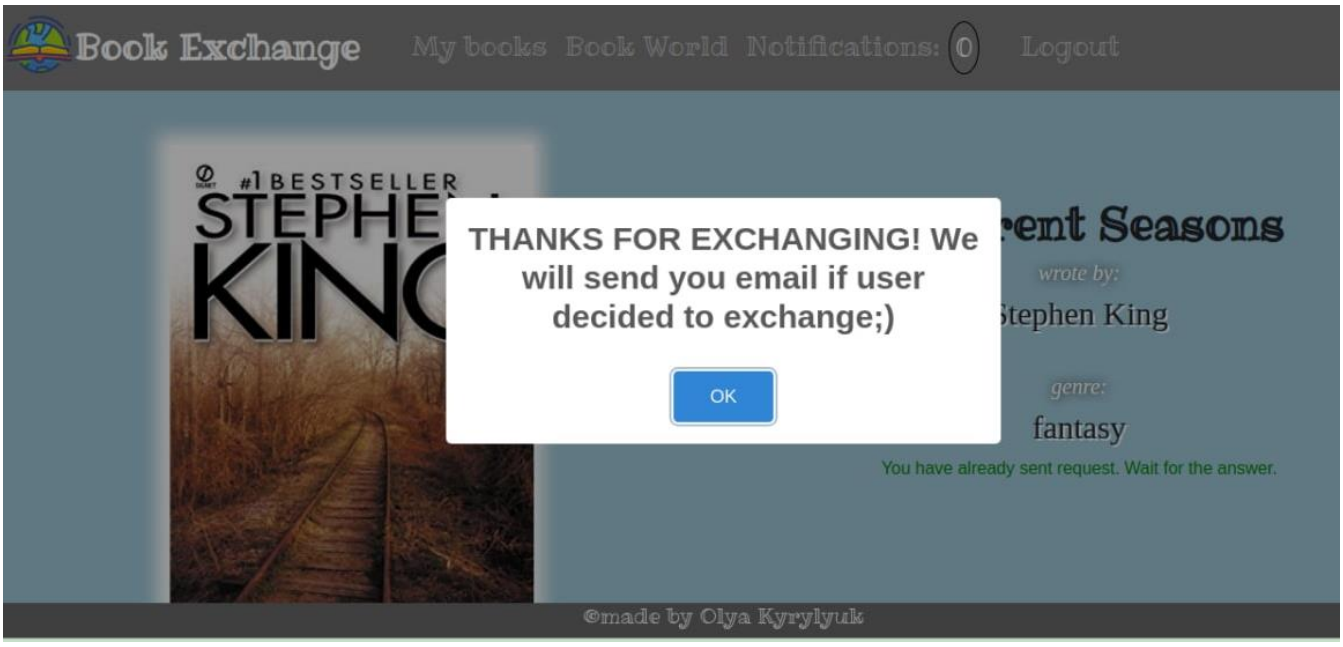


Рисунок 3.23 – Повідомлення про успішно відправлений запит на обмін

Після цього в користувача буде відображатися замість кнопки текст зеленим кольором.

Результат наведено на рисунку 3.24:

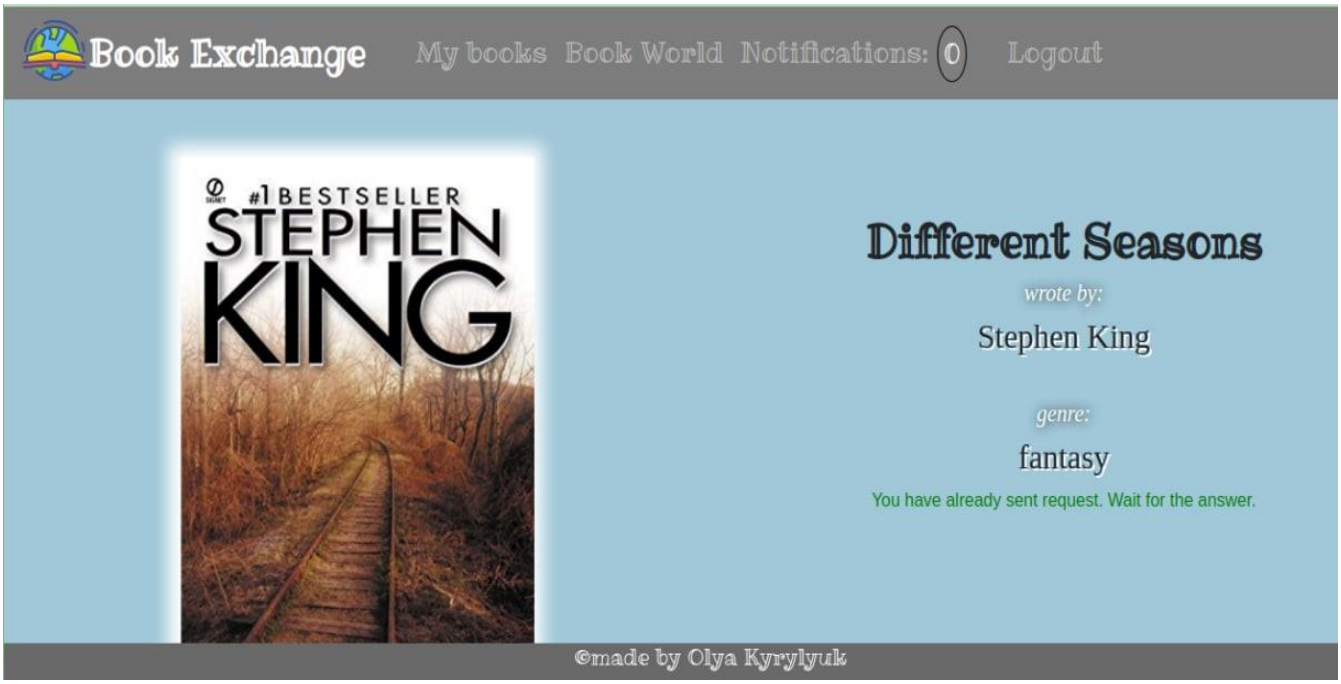


Рисунок 3.24 – Текст повідомлення

Для перегляду обміну книгами потрібно натиснути на вкладку меню, сторінку якої наведено на рисунку 3.25:

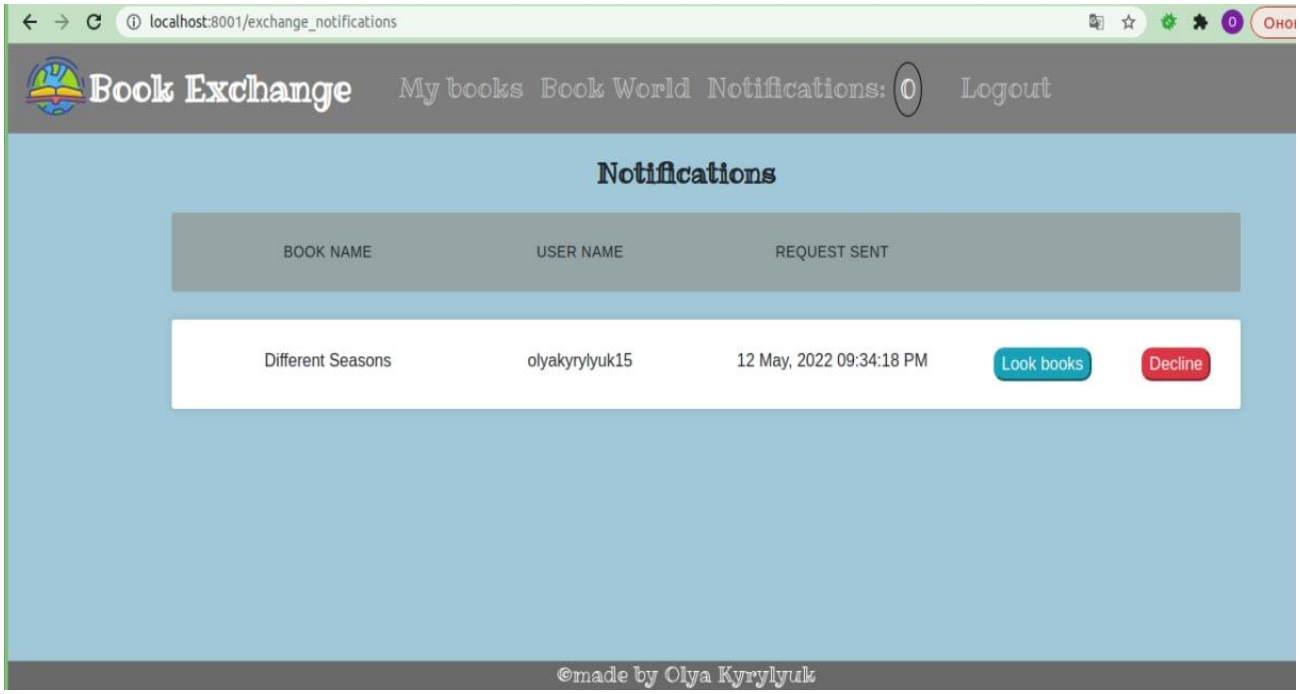


Рисунок 3.25 – Сторінка з переглядом запитів на обмін

Для виходу з системи необхідно натиснути «Login» на вкладці меню.

3.4 Вимоги до технічних та програмних засобів

Для розгортання веб-сайту потрібен хостинг з наступними параметрами:

– об’єм дискового простору – 1 ГБ;
– кількість доменів – 1;
– доступ до мережі Інтернет;
– PhpStorm;
– OpenServer;
– наявність електронної адреси (необхідно для авторизації).

3.5 Розгортання та встановлення системи

Для розгортання та встановлення системи необхідно мати встановлені технології:

– PHP;
– Laravel;
– Composer;
– MySQL;
– PhpMyAdmin.

Оскільки веб-додаток розміщений на системі контролю версій – GitHub, можна склонувати проект ввівши в терміналі:

git clone https://github.com/OlyaKyrylyuk/BookExchange.git

Тоді проект з’явиться на пристрої. Далі необхідно відкрити проект в середовищі розробки програмного забезпечення. Після, виконати команди для встановлення необхідних пакетів:

– npminstall;
– nom run dev;

– composerinstall;

– php artisan key:generate;

– php artisan migrate.

Далі при введенні команди php artisan serve запуститься веб-додаток.

У даному розділі було розроблено базу даних, створена інструкція для користувача, описані вимоги для технічних та програмних засобів, а також прописана інструкція для розгортання та встановлення системи.

4. ТЕСТУВАННЯ ПРОГРАМНОЇСИСТЕМИ

4.1 Вибір та обґрунтування методів тестування додатку

Щоб розробити програмне забезпечення з бездоганними характеристиками, якістю та функціональністю, для інженерів-програмістів дуже важливо проводити суворе та ретельне тестування на ранніх етапах розробки програмного забезпечення. Крім того, тестувальники повинні зосередитися на тестуванні навіть найменших компонентів програмного забезпечення, щоб перевірити його ефективність, а також продуктивність. Однією з таких методик тестування, яка допомагає їм у перевірці якості окремих функцій програмного забезпечення, є модульне тестування.

У програмному продукті блок – це найменший компонент, який перевіряють тестувальники програмного забезпечення. Вони мають набір специфікацій, які допомагають створити очікувану функціональність та продуктивність програмного забезпечення. Коли ці блоки об’єднуються та інтегруються разом, утворюється модуль, який є повністю сформованим компонентом когерентного коду, який можна перевірити, виконавши деякі функціональні сигнатури з численними стимулами. Ці модулі є основною частиною програмного забезпечення і тестуються за допомогою модульного тестування.

Тестування модулів – це різновид техніки тестування «білого ящика» – це процес тестування окремих компонентів, класів, підпрограм, підпрограм або процедур у програмі. Цей тип тестування програмного забезпечення в основному проводиться групою тестувальників програмного забезпечення, які зосереджені на тестуванні програмних компонентів ізольовано. Також відоме як тестування компонентів і програм, модульне тестування збільшує ймовірність виявлення та усунення дефектів, оскільки невеликі частини програмного забезпечення тестуються окремо, що дозволяє ретельно вивчити програму.

Метою модульного тестування є не перевірка функціональності чи продуктивності програмного забезпечення, а пошук і усунення дефектів і помилок у програмному забезпеченні, що ще більше покращує якість всього програмного забезпечення.

Оскільки модулі є невід'ємною частиною програмного забезпечення, їх тестування стає надзвичайно важливим для визначення якості та ефективності програмного забезпечення. Окрім тестування модулів, команда також реалізує модульне тестування для окремого тестування різних компонентів програмного забезпечення. Інші причини тестування модулів:

− причина використання модульного тестування – це спосіб управління комбінованими елементами програмного забезпечення;

− метою є не перевірка функціональності цих модулів, а пошук дефектів і помилок у програмному забезпеченні;

− допомагає керувати складністю тестування програмного забезпечення;

− оскільки він підтримує паралельне тестування, різні модулі можна тестувати одночасно.

4.2 Розробка тестів

Для створення тестів необхідно виконати команду в консолі:

phpartisanmake:testUserTest –unit

Приклад тесту:

```
<?php
namespace Tests\Unit;
use Tests\TestCase;
class UserTest extends TestCase
{
    /**
     * A basic test example.
     *
     * @return void
     */

    public function testExample()
    {
        $this->visit('/')
            ->click('Авторизація')
            ->seePageIs('/login');
```

```
    }
}
```

Результат тесту зображений на рисунку 4.1:

```
d:\OSPanel\domains\Court>vendor\bin\phpunit
PHPUnit 6.5.14 by Sebastian Bergmann and contributors.

...                                                                 3 / 3 (100%)

Time: 597 ms, Memory: 14.00MB

OK (3 tests, 6 assertions)
```

Рисунок 4.1 – Результат тесту

В даному розділі розглядалося тестування розробленого програмного продукту, в ході якого було з’ясовано, що додаток відповідає встановленим вимогам та характеристикам.

ВИСНОВКИ

Отже, після завершення виконання дипломного проекту, результатом є веб-додаток пошуку книг та обміну ними, який відповідає поставленому завданню та усім визначеним вимогам.

На початку виконання проекту було здійснено аналіз подібних рішень, розглянуто веб-сайти інших країн, що стосуються пошуку та обміну книг (BookMooch, PaperBackSwap, BookSwap, Readitswapit), визначено їх основні переваги та недоліки для визначення вимог до пректу.Для реалізації використано наступні технології:

– HTML;
– CSS;
– Bootstrap;
– JQuery;
– JavaScript;
– PHP;
– Laravel.

Основні функціональні можливості веб-додатку:

– реєстрація на сайті з кодуванням паролей в базі даних для безпеки;
– авторизація на сайті;
– перегляд всіх книг на сторінці для неавторизованого користувача;
– перегляду всіх книг для авторизованого користувача з додатковими зручностями;
– перегляд однієї книги;
– створення книги;
– перегляд власних книг;
– перегляд бібліотеки книг на сайті;
– створення запиту на обмін книги;
– створення обміну книги;

– відправлення листа про успішний обмін з контактами користувачів;

– перегляд запитів на обмін книгами;

– вихід з системи.

Основною перевагою даного додатку є можливість знайти книгу для обмінута та зекономити кошти.

Проведено дослідження предметної області на основі існуючих подібних рішень, за допомогою цього вдалося коректно розробити структуру бази даних та реалізувати програмний продукт.

Проведено тестування розробленого додатку задля підтвердження надійності роботи застосунку, результати якого є задовільними. Також проведено дослідження ринку збуту для програмного продукту і обґрунтовано економічну доцільність його розробки.

Отже, представлений веб-додаток успішно сконструйований та готовий до використання. Судячи з цього, можна зробити висновок, що програмний продукт «Book Exchange» є цілком успішним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Репозиторій дипломного проекту в системі контролю версій GitHub

Посилання: https://github.com/OlyaKyrylyuk/BookExchange

2. Карвін Б. Програмування баз даних SQL. Типові помилки та їх усунення. Ранок, 2012 – 336 ст.

3. Matt Stauffer. Laravel: Up and Running: A Framework for Building Modern PHP Apps. BookChef, 2016 – 795 p.

4. FrancescoMalatesta. Learning Laravel's Eloquent. Paperback, 2015 – 202 p.

5. Документація по Laravel(фреймворк)

Посилання: https://laravel.com/docs/5.7

6. Laravel Tutorial (підручник)

Посилання: https://www.tutorialspoint.com/laravel/

7. Документація з модульного тестування

Посилання: https://www.professionalqa.com/module-testing

8. Документація з правильного оформлення дипломного проекту.

Посилання:http://lib.khnu.km.ua/EL_LIBRARY/vidavn/metod/mtd2020_2/73/index.pdf

9. Документація з створеннядинамічного пошуку в Laravel:

Посилання:

https://www.webslesson.info/2018/04/live-search-in-laravel-using-ajax.html

10. Документація з модульного тестування:

Посилання: https://www.techtarget.com/searchsoftwarequality/definition/unit-testing

11. Документація з використанняGmailв Laravel:

Посилання:

https://www.itsolutionstuff.com/post/laravel-8-send-mail-using-gmail-smtp-serverexample.html

12. Документація з використання стилів на сторінці:

Посилання: https://www.w3schools.com/css/

13. Документація з використання LaravelEloquent:

Посилання: https://laravel.com/docs/9.x/eloquent

14. Документація з використання MySQL:

Посилання: https://www.php.net/manual/ru/book.mysql.php

15. Документація з використання phpMyAdmin:

Посилання: https://www.phpmyadmin.net/

ДОДАТОК А

(Обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

1 Загальні відомості.

1.1 Найменування замовника та розробника.

У рамках реалізації дипломного проекту для предмету заплановано розробку веб-додатку «Book Exchange», який дозволить легко, швидко та зручно обмінюватися книгами.

Замовником розробки визначено виконавця проекту – кафедру інженерії програмного забезпечення.

Розробник програмного забезпечення - студентка групи ІПЗс-19-1 Кирилюк Ольга.

1.2 Термін надання послуг.

Даний програмний продукт потрібно розробити протягом чотирьох місяців з 11.02.2022 по 11.06.2022. Кожного тижня, в п`ятницю починаючи з 11.09.2022 року потрібно демонструвати альфа-версії веб-додатку. В разі невиконання умов контракт вважається анульованим.

1.3. Джерело та порядок фінансування.

Розробка системи фінансується замовником у рамках виконання проекту. Сума проекту – $600(шістсот доларів). Якщо проект буде реалізований невчасно, сума проекту зменшиться в залежності від терміну доопрацювання програмного забезпечення після встановленого терміну.

1.4. Порядок оформлення та пред’явлення замовнику результатів.

Постачальник після завершення розробки програмного продукту передає замовнику результати розробки, а саме:

– акт приймання-передачі наданих послуг (у двох екземплярах);

– програмне забезпечення (у одному екземплярі).

2. Вимоги до програмного продукту.

2.1 Вимоги до програмного продукту в цілому.

Даний веб-додаток повинен реалізувати можливість пошуку користувачем книг та обмін між ним та іншим користувачем, з метою економії часу, коштів, простору в кімнаті та екології в цілому.

2.2 Вимоги користувача та системні вимоги.

Вимоги користувача:

– зручність використання та простота сайту;

– інтуїтивно - зрозумілий;

– зручна система навігації для перегляду великої кількості справ.

Далі складаються системні вимоги. Вони включать в себе:

2.3 Вимоги до архітектури системи. Так як ПП є веб-додатком хостинг повинен підтримувати наступні вимоги:

– RAM – 512 Мб;

– місце на NVMe-диску - до 3 Гб;

– підтримка Apache, OpenLiteSpeed;

– підтримка PHP 5.2 - 8.0;

– антивірусний захист;

– резервне копіювання;

– доступ до PHPMyAdmin;

– віддалений доступ до MySQL.

2.4 Вимоги до параметрів обладнання.

– наявність веб-браузера;

– можливість підключення до мережі Інтернет.

2.5 Вимоги до програмного інтерфейсу.

– веб-додаток повинен буди адаптивним, зручним для використання на будь-якому пристрої;

– повинен бути відображений логотип, який відповідає назві сайту – «Book Exchange».

– кольорова палітра програмного продукту – світло голубий та сірий кольори;

– відтінки повинні бути приємними при поєднанні на сайті;

– натуральні кольори більш сприймаються очима, чим штучні. Тому кольора повинні бути досить спокійними і не відштовхувати користувача;

– шрифт основного тексту – за бажанням;

– контент веб-додатку реалізований англійською мовою для того щоб користувач з будь-якої країни мав можливість придбати та використовувати бажаний товар;

– на основній сторінці можливо переглядати лише декілька сторінок;

– буде створена пагінація, що дозволяє переходити по сторінкам та переглядати книги зручно;

– оформлення сайту: сучасний мінімалізм;

– дизайн сайту та головна сторінка повинні не відлякувати, а залучати відвідувачів.

3. Вимоги до структури системи.

Система повинна бути відкритою, тобто будь-який користувач може користуватися сайтом, потрібно лише зареєструватися, або авторизуватися якщо користувач попередньо був зареєстрований в веб-додатку “Book Exchange”.

Завдяки гнучкості програмного продукту можна модифікувати або додати нові функції та можливості, без порушень цілісності системи.

Сайт повинен коректно відображатися на різних пристроях (ноутбук, телефон, комп`ютер, планшет, та ін.)

1. Функціональні вимоги.

При вході на сайт повинна бути сторінка, доступна для неавторизованого гостя. На цій сторінці є посилання на:

– реєстрацію, – якщо користувач вперше користується сайтом;

– авторизацію, – якщо користувач попередньо зареєструвався.

На сторінці реєстрації поля:

– username;

– email;

– phone;

– password;

– confirm password;

– кнопка для реєстрації.

На сторінці авторизації поля:

– email;

– password;

– можливість ввійти в систему для обміну книгами в подальшому.

Гостьова сторінка містить меню, пункти якого:

– реєстрація;

– авторизація;

– інформаційна сторінка для гостя.

Також з лівої сторони меню повинен відображатися логотип.

Сторінка користувача містить меню, пункти якого:

– сторінка перегляду власних книг;

– додати книгу;

– перегляд всіх книг для пошуку бажаної та обміном;

– перегляд запитів на обмін;

– вихід з системи з правої сторони меню.

Сторінка додати книгу складається з:

– назви книги;

– автора;

– картинки;

– вибору жанру книги.

Сторінки з усіма книгами повинні містити пагінацію, на одній сторінці повинно відображатися не більше 12 книг.

2. Вимоги до надійності.

Програма має перевіряти вхідні дані під час реєстрації користувача на правильність введення пошти, довжини введеного паролю і підтвердження паролю для правильності введених даних користувачем, який вирішив зареєструватися.

3. Вимоги до складу і параметрів технічних засобів.

Необхідна електронна обчислювальна машина з можливістю виведення та вводу інформації, наприклад: при використанні персональних комп’ютерів необхідна наявність пристроїв введення

– клавіатура;

– миша.

і виводу (монітор) з наявністю підключення до мережі Інтернет. Оскільки в сучасному світі Інтернетом користується весь світ, це не є проблемною вимогою.

4. Вимоги до інформаційної та програмної сумісності.

Дана веб-система може використовуватись на комп’ютерах із будь-якою встановленою операційною системою, на якій встановлений веб-браузер. ТА

Сайт реалізовуватиметься за допомогою технологій:

– PHP;
– Laravel;
– JS;
– HTML;
– CSS;
– Bootstrap;
– jQuery;
– MySQL;
– та інші.

5. Вимоги до маркування й упакування.

Готовий проект повинен бути розміщений на системі контролю версій GitHub, а копія проекту поставляється на Flesh носієві.

6. Вимоги до транспортування і зберігання.

Повинні дотримуватися правила експуатації Flesh-носія, а саме:

Забороняється:

– примусовий витяг накопичувача з порту USB до завершення будь-якої виконуваної операції або примусове переривання операції;

– спроби форматування накопичувача не в тій файлової системи, яка існувала в початковому стані пристрою в момент його придбання (робота з довільно обраної файлової системою може бути не передбачена в структурі контролера флеш-пам’яті);

– виконання з цікавості тестової операції запису у флеш-пам’ять, після чого накопичувач не видаляється з системи за допомогою механізму безпечного

вилучення (виправлення недоліку - повторне форматування пристрою в вихідної файлової системи або використання режиму форсованої зупинки);

– після безпечного видалення пристрій з флеш-пам’яттю не впізнав в файловому менеджері і спеціалізованих програмах (причина нестачі - було виконано безпечне видалення накопичувача, але пристрій не було своєчасно вилучено з порту USB; виправлення браку - остаточне вилучення пристрою з порту USB).

7. Вимоги, щодо взаємодії та інтеграції з іншими системами.

Вся інформація стосовно справ зберігається в базі даних.

8. Стадії та етапи розробки програмногозабезпечення для реалізації електронного гаманця криптовалюти Tronix з реалізацією інтерфейсу у вигляді Telegram-бота подані у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.22 – 31.01.22	Обґрунтування необхідності розробки програми	Характеристика та опис ПЗ; підстави для розробки і призначення ПЗ; функціональні вимоги до розроблюваної системи; порядок контролю і приймання ПЗ.
Ескізний проект 01.02.22 – 14.02.22	Розробка ескізного проекту	Визначення структури вхідних і вихідних даних; попередній вибір технологій; визначення потрібних алгоритмів.
Технічний проект 15.02.22 – 28.02.22	Розробка технічного проекту	Затвердження структури вхідних і вихідних даних; розробка алгоритмів; розробка структури програми; вибір технологій.

Продовження таблиці А.1

1	2	3
Робочий проект 01.03.22 – 10.04.22	Розробка програмного забезпечення	Реалізація програмного забезпечення; виправлення помилок; тестування.
Розробка програмної документації 11.04.22 – 20.04.22	Розробка документації для програмного забезпечення	Розробка документації користувача, інструкції по запуску та зупинці ПЗ
Тестування системи 21.04.22 – 30.04.22	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Здача проекту	Здача програми замовнику	Передача вихідного коду та документації замовнику

ДОДАТОК Б

(Обов’язковий)

ДІАГРАМИ

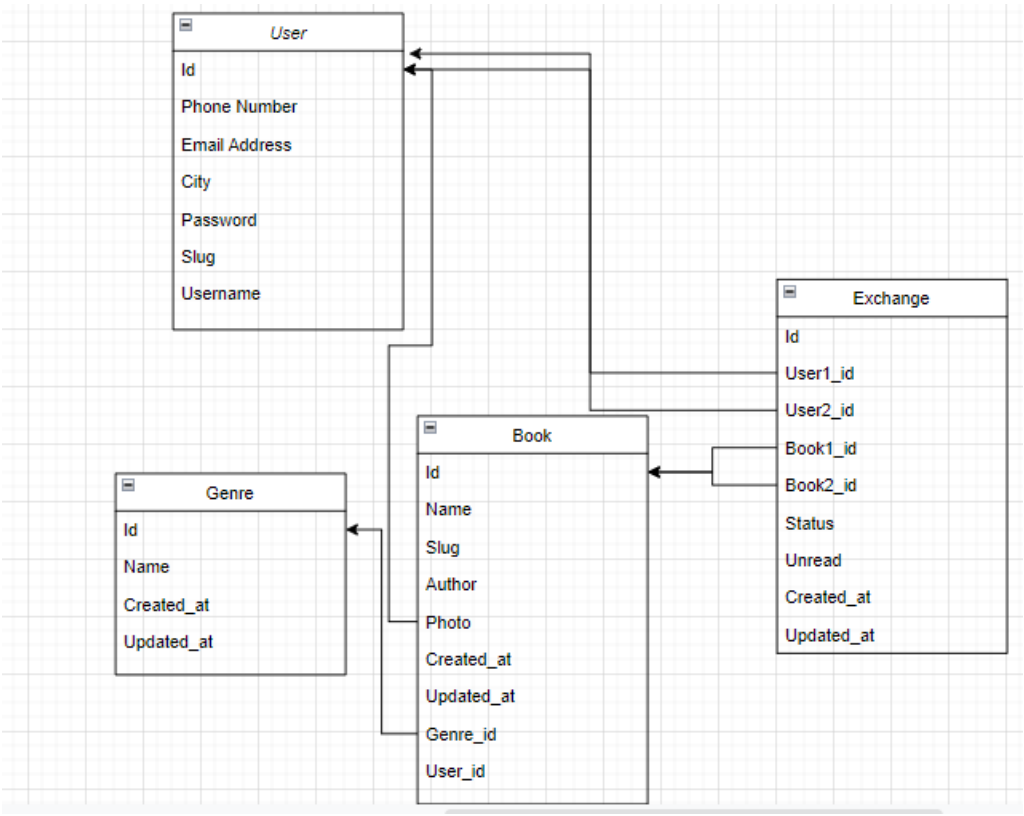


Рисунок Б.1 – Схема бази даних

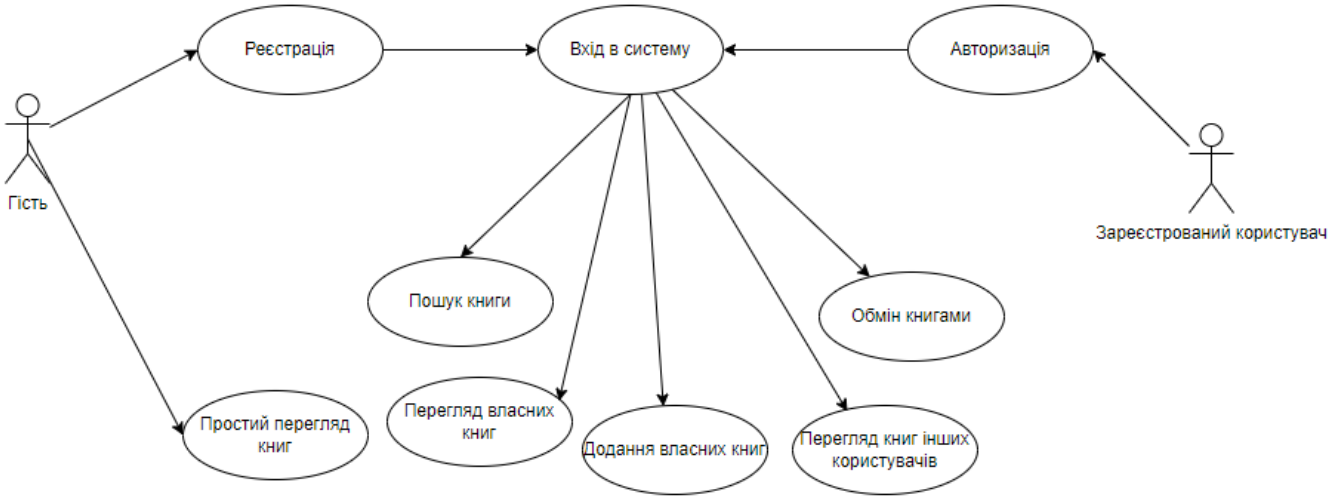


Рисунок Б.2 – Діаграма варіантів використання

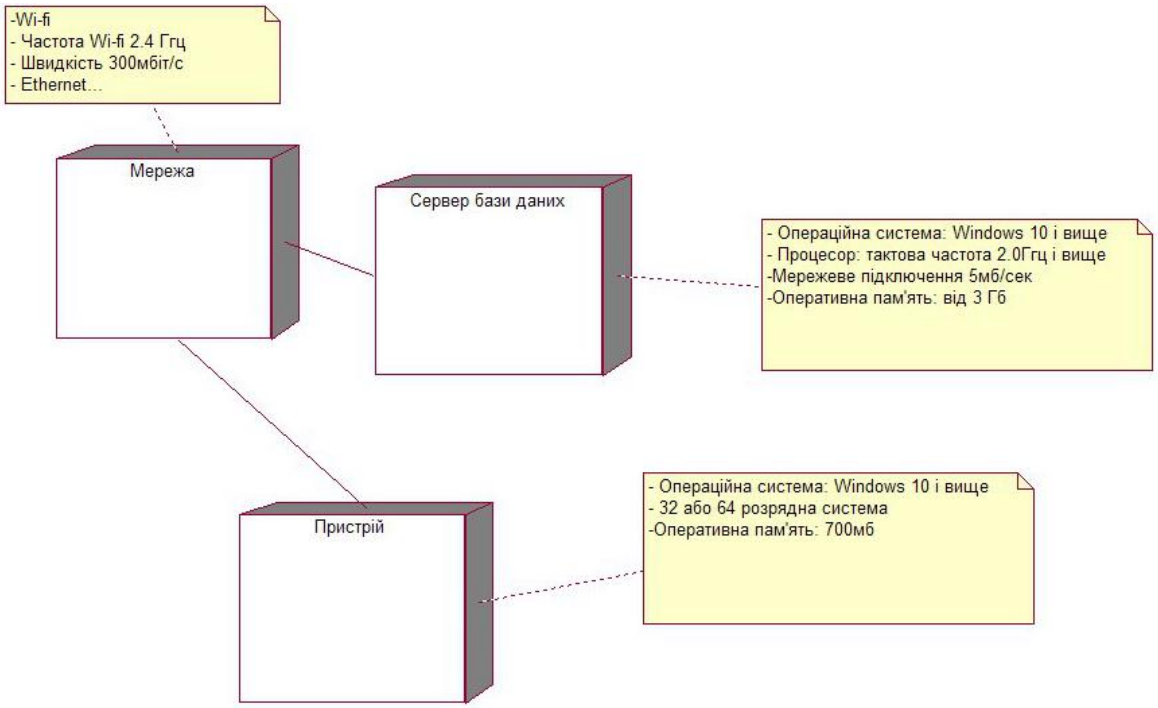


Рисунок Б.3 – Діаграма розгортання

ДОДАТОК В

(Обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

create_users_table.php

```
<?php
        use Illuminate\Database\Migrations\Migration;
        use Illuminate\Database\Schema\Blueprint;
        use Illuminate\Support\Facades\Schema;
        class CreateUsersTable extends Migration
        {
            /**
             * Run the migrations.
             *
             * @return void
             */
            public function up()
            {
                Schema::create('users', function (Blueprint $table) {
        $table->id();
                    $table->string('username')->required();;
                    $table->string('phone');
                    $table->string('city');
                    $table->string('email')->unique()->required();;
        $table->string('password');
                    $table->string('slug')->required();
                    $table->timestamp('email_verified_at')->nullable();
        $table->rememberToken();
                    $table->timestamps();
                });
            }
            /**
             * Reverse the migrations.
             *
             * @return void
             */
            public function down()
            {
                Schema::dropIfExists('users');
            }
        }
```

create_genres_table.php

```
<?php
        use Illuminate\Database\Migrations\Migration;
        use Illuminate\Database\Schema\Blueprint;
        use Illuminate\Support\Facades\Schema;
        class CreateGenresTable extends Migration
        {
            /**
             * Run the migrations.
             *
             * @return void
             */
            public function up()
            {
                Schema::create('genres', function (Blueprint $table) {
        $table->id();
                    $table->string('name')->required();
                    $table->timestamps();
                });
            }
            /**
             * Reverse the migrations.
             *
             * @return void
             */
            public function down()
            {
                Schema::dropIfExists('genres');
            }
        }
```

create_books_table.php

```
<?php
        use Illuminate\Database\Migrations\Migration;
        use Illuminate\Database\Schema\Blueprint;
        use Illuminate\Support\Facades\Schema;
        class CreateBooksTable extends Migration
        {
            /**
             * Run the migrations.
             *
             * @return void
             */
            public function up()
            {
                Schema::create('books', function (Blueprint $table) {
        $table->id();
                    $table->string('name')->required();
                    $table->string('slug')->unique();
```

```
            $table->string('author')->required();
            $table->string('photo')->nullable();
            $table->bigInteger('user_id')->unsigned()->nullable();
            $table->foreign('user_id')->references('id')->on('users')->onDelete('set null');
            $table->bigInteger('genre_id')->unsigned()->nullable();
            $table->foreign('genre_id')->references('id')->on('genres')->onDelete('set null');
        $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('books');
    }
}
```

Create_exchanges_table.php

```
<?php
    use Illuminate\Database\Migrations\Migration;
    use Illuminate\Database\Schema\Blueprint;
    use Illuminate\Support\Facades\Schema;
    class CreateExchangesTable extends Migration
    {
        /**
         * Run the migrations.
         *
         * @return void
         */
        public function up()
        {
            Schema::create('exchanges', function (Blueprint $table) {
        $table->id();
            $table->bigInteger('user1_id')->unsigned()->nullable();
            $table->foreign('user1_id')->references('id')->on('users')->onDelete('cascade');
            $table->bigInteger('user2_id')->unsigned()->nullable();
            $table->foreign('user2_id')->references('id')->on('users')->onDelete('cascade');
            $table->bigInteger('book1_id')->unsigned()->nullable();
            $table->foreign('book1_id')->references('id')->on('books')->onDelete('cascade');
            $table->bigInteger('book2_id')->unsigned()->nullable();
            $table->foreign('book2_id')->references('id')->on('books')->onDelete('cascade');
            $table->string('status')->default('');
        $table->boolean('unread')->default(1);
            $table->timestamps();
```

```
            });
        }
        /**
         * Reverse the migrations.
         *
         * @return void
         */
        public function down()
        {
            Schema::dropIfExists('exchanges');
        }
    }
```

UserController.php

```
<?php
    namespace App\Http\Controllers;
    use Illuminate\Support\Facades\DB;
    use App\Models\Book;
    use App\Models\User;
    use Illuminate\Http\Request;
    class UserController extends Controller
    {
        public function get_profile($slug){
    //        $profile = Book::with('users','genre')->get();
    //        $profile = User::where('slug', $slug)->with('books')->firstOrFail();
            $profile = User::with(['books' => function($q){
    $q->with('genre');
            }])
                ->where('slug', $slug)
                ->firstOrFail();
            return view('users.profile',['profile'=>$profile]);
    }
    }
```

BookController.php

```
<?php
    namespace App\Http\Controllers;
    use App\Models\Genre;
    use App\Models\Book;
    use App\Models\Exchange;
    use App\Models\User;
    use App\Services\BookService;
    use Illuminate\Http\Request;
    use App\Events\SuccessExchange;
    class BookController extends Controller
    {
        private $bookService;
        public function __construct(BookService $bookService)
    {
```

```
        $this->bookService = $bookService;
    }
    public function home()
    {
        $user = auth()->user();
        $books = Book::with(['genre', 'user'])->where('user_id', $user->id)-
>orderBy('created_at', 'desc')->get();
        return view('home', ['books'=>$books]);
}
public function index(){
        $books = Book::with(['genre', 'user'])->orderBy('created_at', 'desc')->get();
        return view('books.index', ['books'=>$books]);
}
    public function show($slug)
    {
       $data =  $this->bookService->showOneBook($slug);
$book = $data[0];
        $current_user = $data[1];
        $exchange = $data[2];
        return view('/books/single_book',['book'=>$book, 'current_user'=>$current_user,
'exchange'=> $exchange]);
}
public function create(){
        $genres = Genre::all();
        return view('books.create', ['genres'=>$genres]);
}
    public function store(Request $request)
{
        $input = $request->except(['genres']);
        $this->bookService->saveBook($input, $request->genres, $request->hasFile('photo'),
$request->name, $request->photo);
        return redirect()->route('books.index');
}
    public function bookRequest($slug)
    {
        $this->bookService->sendRequest($slug);
return redirect()->back()->with('message', 'IT WORKS!');
}
    public function choose_exchange_book($id)
{
        $exchange = Exchange::where('id', $id)->first();
        $user_profile = User::with('books')->where('id', $exchange->user1_id)->get();
        return view('books.choose_exchange_book', ['user_profile'=>$user_profile,
'exchange'=>$exchange]);
}
    public function accept_exchange_book($id, $book_id)
{
        $exchange = Exchange::where('id', $id)->first();
```

```
        $exchange->book2_id = $book_id;
$exchange->status = 'success';
        $exchange->save();
        $user_from = User::where('id',$exchange->user1_id)->first();
        $book1 = Book::where('id', $exchange->book1_id);
        $book2 = Book::where('id', $exchange->book2_id);
        event(new SuccessExchange(auth()->user(), $user_from, $book1, $book2));
return redirect('/home')->with('message', 'Request was sent to user! Wait for the message on
your email about applying or declining request!');
}
    public function welcome()
    {
        $books = Book::with(['genre', 'user'])->orderBy('created_at', 'desc')->get();
        return view('books.welcome', ['books'=>$books]);
}
}
```

NotificationsController.php

```
<?php
        namespace App\Http\Controllers;
        use App\Models\Book;
        use App\Models\Exchange;
        use App\Models\User;
        use Illuminate\Http\Request;
        class NotificationsController extends Controller
        {
            public function lookRequests()
            {
                $current_user = $user = auth()->user();
        $user_from = [];
                $request_book = [];
                $exchanges = Exchange::where('user2_id', $current_user->id)->where('status', '')-
        >get();
        if ($exchanges->count()  !=0)
                {
                    $user_from = User::where('id', $exchanges[0]->user1_id)->get();
                    $request_book = Book::where('id', $exchanges[0]->book1_id)->get();
        foreach ($exchanges as $exchange)
                    {
                        $exchange->unread = 0;
                        $exchange->save();
                    }
                }
                return view('notifications.look_requests',['exchanges'=>$exchanges,
        'user_from'=>$user_from, 'request_book'=>$request_book]);
        }
        }
```

User.php

```
<?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\Passport\HasApiTokens;
        use Cviebrock\EloquentSluggable\Sluggable;
        class User extends Authenticatable
        {
            use Notifiable, HasApiTokens;
            use Sluggable;
            /**
             * The attributes that are mass assignable.
        *
             * @var array
             */
        protected $fillable = [
                'username',
                'city',
                'phone',
                'email',
                'password',
            ];
            /**
             * The attributes that should be hidden for arrays.
        *
             * @var array
             */
        protected $hidden = [
                'password', 'remember_token',
            ];
            /**
             * The attributes that should be cast to native types.
        *
             * @var array
             */
        protected $casts = [
                'email_verified_at' => 'datetime',
            ];
            public function books()
            {
                return $this->hasOne(Book::class);
        }
            public function exchanges()
            {
                return $this->hasMany(Exchange::class);
        }
```

```
    public function sluggable()
    {
return [
'slug' => [
                'source' => 'username'
            ]
        ];
    }
}
```

Genre.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Genre extends Model
{
    use HasFactory;
protected $fillable = [
        'name',
    ];
    public function books()
    {
        return $this->hasMany(Book::class);
}
}
```

Book.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Book extends Model
{
    use HasFactory;
protected $fillable = [
        'name', 'photo','slug','author', 'user_id', 'genre_id'
];
    public function user()
    {
        return $this->belongsTo(User::class);
}
    public function genre()
    {
        return $this->belongsTo(Genre::class);
}
    public function exchanges()
    {
        return $this->hasMany(Exchange::class);
```

```
    }
    }
```

Exchange.php

```
<?php
    namespace App\Models;
    use Illuminate\Database\Eloquent\Factories\HasFactory;
    use Illuminate\Database\Eloquent\Model;
    class Exchange extends Model
    {
        use HasFactory;
    protected $fillable = [
            'status','book2_id','book1_id', 'user1_id', 'user2_id'
    ];
        public function user()
        {
            return $this->belongsTo(User::class);
    }
        public function book()
        {
            return $this->belongsTo(Book::class);
    }
    }
```

ExchangeMail.php

```
<?php
    namespace App\Mail;
    use Illuminate\Bus\Queueable;
    use Illuminate\Contracts\Queue\ShouldQueue;
    use Illuminate\Mail\Mailable;
    use Illuminate\Queue\SerializesModels;
    class ExchangeMail extends Mailable
    {
        use Queueable, SerializesModels;
        public $user1;
        public $user2;
        public $book1;
        public $book2;
        /**
         * Create a new message instance.
    *
         * @return void
         */
        public function __construct($user1, $user2, $book1, $book2)
    {
            $this->user1 = $user1;
            $this->user2 = $user2;
            $this->book1 = $book1;
            $this->book2 = $book2;
```

```
    }
    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this
            ->view('email.exchange')
            ->subject('Book exchange was created!')
->with([
                        'user1' => $this->user1,
                        'user2' => $this->user2,
                        'book1' => $this->book1,
                        'book2' => $this->book2
                    ]);
    }
}
```

ExchangeMailFor2User.php

```
<?php
    namespace App\Mail;
    use Illuminate\Bus\Queueable;
    use Illuminate\Contracts\Queue\ShouldQueue;
    use Illuminate\Mail\Mailable;
    use Illuminate\Queue\SerializesModels;
    class ExchangeMailFor2User extends Mailable
    {
        use Queueable, SerializesModels;
        public $user1;
        public $user2;
        public $book1;
        public $book2;
        /**
         * Create a new message instance.
    *
         * @return void
         */
        public function __construct($user1, $user2, $book1, $book2)
    {
            $this->user1 = $user1;
            $this->user2 = $user2;
            $this->book1 = $book1;
            $this->book2 = $book2;
        }
        /**
         * Build the message.
```

```
     *
     * @return $this
     */
    public function build()
    {
        return $this
            ->view('email.exchange2')
            ->subject('Book exchange was created!')
->with([
                        'user1' => $this->user1,
                        'user2' => $this->user2,
                        'book1' => $this->book1,
                        'book2' => $this->book2
                    ]);
    }
}
```

BookService.php

```
<?php
namespace App\Services;
use App\Models\Book;
use App\Models\Exchange;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Str;
class BookService
{
    public function saveBook($input, $genres, bool $hasFile, $name, $photo)
{
        $input['user_id'] = Auth::user()->id;
$input['genre_id'] = $genres;
        $input['slug'] = Str::slug($name, '-');
$newPhotoName = "";
if($hasFile) {
            $newPhotoName = time() . '-' . $name ."." . $photo->extension();
            $photo->move(public_path('images/books'), $newPhotoName);
}
        $input['photo'] =  $newPhotoName;
        Book::create($input);
    }
    public function sendRequest($slug)
    {
        $current_user = $user = auth()->user();
        $book_request = Book::with('user')->where('slug', $slug)->get();
Exchange::create([
            'user1_id' => $current_user->id,
            'user2_id' => $book_request[0]->user->id,
            'book1_id' => $book_request[0]->id,
]);
```

```
    }
    public function showOneBook($slug)
    {
        $current_user = $user = auth()->user();
        $book = Book::with('user')->where('slug',$slug)->first();
$exchange = Exchange::with(
            ['book' => function($q) use ($book){
                $q->where('id', $book->id);
}
            ],
            ['user' => function($q) use ($current_user){
                $q->where('id', $current_user->id );
}
            ])->where('user1_id', $current_user->id)
->get();
//        $exchange = Book::with(['exchanges', 'user'=>function($q) use ($current_user){
//                $q->where('id', $current_user->id );
//            }])->get();
//        dd($exchange);
        return [$book, $current_user, $exchange];
}
}
```

SuccessEvent.php

```
<?php
    namespace App\Events;
    use Illuminate\Broadcasting\Channel;
    use Illuminate\Broadcasting\InteractsWithSockets;
    use Illuminate\Broadcasting\PresenceChannel;
    use Illuminate\Broadcasting\PrivateChannel;
    use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
    use Illuminate\Foundation\Events\Dispatchable;
    use Illuminate\Queue\SerializesModels;
    class SuccessExchange
    {
        use Dispatchable, InteractsWithSockets, SerializesModels;
        /**
         * Create a new event instance.
    *
         * @return void
         */
        public $user1;
        public $user2;
        public $book1;
        public $book2;
        public function __construct($user1, $user2, $book1, $book2)
    {
            $this->user1 = $user1;
```

```
        $this->user2 = $user2;
        $this->book1 = $book1;
        $this->book2 = $book2;
    }
    /**
     * Get the channels the event should broadcast on.
*
     * @return \Illuminate\Broadcasting\Channel|array
*/
    public function broadcastOn()
    {
        return new PrivateChannel('channel-name');
}
}
```

Web.php

```
<?php
use App\Http\Controllers\BookController;
use App\Http\Controllers\NotificationsController;
use App\Http\Controllers\PassportAuthController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/
Route::post('register', [PassportAuthController::class, 'register']);
Route::post('login', [PassportAuthController::class, 'login']);
Route::middleware('auth:api')->group(function () {
    Route::get('get-user', [PassportAuthController::class, 'userInfo']);
});
Route::get('/', [BookController::class, 'welcome']);
Route::get('/home', [BookController::class, 'home'])->middleware('auth');
//книги
Route::resource('/books', BookController::class)->middleware('auth');
Route::get('/books/{slug}/exchange_request', [BookController::class, 'bookRequest'])-
>name('exchange.request');
//повідомлення
Route::get('/exchange_notifications', [NotificationsController::class, 'lookRequests'])-
>name('exchange.request');
Route::get('/exchanges/{id}/choose_exchange_book', [BookController::class,
'choose_exchange_book'])->middleware('auth');
```

```
Route::post('/exchanges/{id}/accept/{book_id}', [BookController::class,
'accept_exchange_book'])->middleware('auth');
//Route::post('/books/{slug}/order', BookController::class);
//Route::get('/books', [BookController::class, 'get_all_books'])->middleware('auth');
//профіль
Route::get('/profile/{slug}', [UserController::class, 'get_profile'])->middleware('auth');
```

Register.blade.php

```
@extends('templates.main')
                            @section('content')
                            <div class="container mt-5 mp-5" style="padding-bottom: 55px;">
                            <div class = "white_block">
                            <div>
                            <img class="imgLogin" src="{{ asset('images/photo.jpg') }}" />
                            </div>
                            <div class="formRegisterLogin">
                            <h1>Register Page</h1>
                            <form method="POST" action="{{ route('register') }}">
                            @csrf
                            <div class="form-group">
                            <label for="exampleInputEmail1">Username</label>
                            <input type="text" class="form-control" name="username"
                            placeholder="Enter username">
                            @error('username')
                            <span class="invalid-feedback" role="alert">
                            <strong>{{$message}}</strong>
                            </span>
                                                @enderror
                            </div>
                            <div class="form-group">
                            <label for="exampleInputEmail1">Phone</label>
                            <input type="text" class="form-control" name="phone"  placeholder="Enter
                            phone">
                            @error('phone')
                            <span class="invalid-feedback" role="alert">
                            <strong>{{$message}}</strong>
                            </span>
                                                @enderror
                            </div>
                            <div class="form-group">
                            <label for="exampleInputEmail1">City</label>
                            <input type="text" class="form-control" name="city" placeholder="Enter
                            city">
                            @error('city')
                            <span class="invalid-feedback" role="alert">
                            <strong>{{$message}}</strong>
                            </span>
```

```
                    @enderror
</div>
<div class="form-group">
<label for="exampleInputEmail1">Email address</label>
<input type="email" class="form-control" name="email" aria-
describedby="emailHelp" placeholder="Enter email">
@error('email')
<span class="invalid-feedback" role="alert">
<strong>{{$message}}</strong>
</span>
                    @enderror
</div>
<div class="form-group">
<label for="exampleInputPassword1">Password</label>
<input type="password" class="form-control" name="password"
placeholder="Password">
@error('password')
<span class="invalid-feedback" role="alert">
<strong>{{$message}}</strong>
</span>
                    @enderror
</div>
<div class="form-group">
<button type="submit" class="btn btn-primary">{{ __('Register')
}}</button>
</div>
</form>
</div>
</div></div>
@endsection
```

Login.blade.php

```
@extends('templates.main')
                    @section('content')
                    <div class="container" style ="padding-bottom: 35px;">
                    <div class="container mt-5 mp-5">
                    <div class = "white_block">
                    <div>
                    <img class="imgRegiserLogin" src="{{ asset('images/photo.jpg') }}" />
                    </div>
                    <div class="formRegisterLogin">
                    <h1>Login Page</h1>
                    <form method="POST" action="{{ route('login') }}">
                    @csrf
                    <div class="form-group">
                    <label for="exampleInputEmail1">Email address</label>
                    <input type="email" name="email" class="form-control"
```

```
id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter
email">
@error('email')
<span class="invalid-feedback is-invalid" role="alert">
<strong>{{$message}}</strong>
</span>
                        @enderror
</div>
<div class="form-group">
<label for="exampleInputPassword1">Password</label>
<input type="password" name="password" class="form-control"
id="exampleInputPassword1" placeholder="Password">
@error('password')
<span class="invalid-feedback" role="alert">
<strong>{{$message}}</strong>
</span>
                        @enderror
</div>
<div class="form-group">
<button type="submit" class="form-control btn btn-primary">Submit</button>
</div>
</form>
<a href="/register">Aren't registered yet? Click here</a>
</div>
</div></div></div>
@endsection
```

Create.blade.php

```
@extends('templates.main')
                        @section('content')
                        <div class="container" style ="padding-bottom: 35px;">
                        <h1>Create Book</h1>
                        <form method="POST" action="{{ route('books.store') }}"
                        enctype="multipart/form-data">
                        {{--    <form  enctype="multipart/form-data" class="form">--}}
                        @csrf
                        <div class="form-group">
                        <label for="exampleInputEmail1">Book name</label>
                        <input type="text" name="name" class="form-control"
                        id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter
                        name">
                        </div>
                        <div class="form-group">
                        <label for="exampleInputEmail1">Author</label>
                        <input type="text" name="author" class="form-control"
                        id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter
                        author">
                        </div>
```

```
<div class="form-group">
<i class="fa fa-cloud"></i>
<label for="picture_input">Photo</label>
<div class="wrapper">
<div class="file-upload">
<input type="file" class="upload" name="photo" id="picture_input"
accept="image/png, image/jpeg, image/jpg"/>
<h1 id = "uploadfile_text" name="uploadfile_text">Choose photo</h1>
<input id="photo_path" type="hidden" name="photo_path" />
<div class="file-upload__image">
</div>
</div>
</div>
</div>
</div>
<div class="form-group">
<label for="exampleInputEmail1">Genre</label>
<select  class="form-control" name="genres">
@foreach($genres as $genre)
<option value= " {{ $genre->id }} "> {{$genre->name}} </option>
@endforeach
</select>
</div>
<div class="form-group">
<button type="submit" class="form-control btn btn-primary">Submit</button>
</div>
</form>
</div>
@endsection
@section('JS')
<script type="text/javascript" >
$(document).ready(function(){
        $('input.upload').on('change', addFile);
        // document.querySelector('.form').addEventListener('submit',()=>{
//
        //     let formData = new
FormData(document.querySelector('.form'))
        //     // formData.set('photo', document.querySelector('.form
.upload').files[0].name);
        //     let base_URL= document.querySelector('.file-upload__image
img').src;
        //     console.log(document.querySelector('.form
.upload').files[0])
        //     console.log(formData.get('photo'))
//
        //     console.log(formData.get('_token'))
//     $.ajax({
//         headers: {
```

```
        //             'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')
//         },
        //         type:'POST',
        //         url:'/books',
        //         processData: false,
        //         contentType: false,
        //         data : formData,
//         success:function(data){
        //             console.log(data);
        //         }
        //     });
        // })
    });
function addFile (event) {
        document.getElementById("uploadfile_text").style.display = 'none';
        var jqDisplay = $('div.file-upload__image');
var reader = new FileReader();
        var selectedFile = event.target.files[0];
reader.onload = function (event) {
            var imageSrc = event.target.result;
            document.getElementById("photo_path").value = imageSrc;
            jqDisplay.html('<img src="' + imageSrc + '"  alt="uploaded
Image" class="uploaded_image">');
};
        reader.readAsDataURL(selectedFile);
    }
</script>
@endsection
```

ДОДАТОК Г

(Обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ


Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
ДИПЛОМНИЙ ПРОЕКТ
на тему:
«Веб-додаток «Book Exchange» для пошуку книг та обміну ними»
Підготувала:
Студентка групи ІПЗс-19-1
Кирилюк Ольга Олександрівна
Керівник дипломного проекту:
Кандидат пед. наук, доцент
Праворська Наталія Іванівна


Мета проекту:
Реалізувати веб-додаток з можливістю обмінюватися книгами з іншими користувачами за допомогою таких технологій як: HTML, CSS, Bootstrap, JavaScript, Laravel, PHP, MySQL.

Актуальність теми проекту:

Попри те, що сучасний світ з кожним роком стає все більш модернізованим і з'являються все нові можливості для зручного читання книг в електронному форматі, паперова книга й досі вважається кращим способом занурення у чарівний світ для книгоманів.

А за допомогою програмних продуктів з'являються нові способи пошуку потрібної книги (особливо якщо вона застаріла та її важко знайти на полицях в магазині) або обміну для тих, хто не має можливості дозволити собі купувати нові книги. Саме тому було обрано дану предметну область для реалізації дипломного проекту.

Завдання проекту:

- ✓ дослідити предметну область;
- ✓ проаналізувати існуючі подібні рішення;
- ✓ визначити функціональні та нефункціональні вимоги;
- ✓ спроектувати структуру та розробити базу даних;
- ✓ розробити реєстрацію та авторизацію;
- ✓ реалізувати можливість додання та перегляд книг;
- ✓ розробити можливість обміну книгами;
- ✓ реалізувати відправку повідомлення про обмін для користувача;
- ✓ протестувати проект на коректність роботи.

Проект розроблений з використанням таких технологій:

Bootstrap

HTML CSS

php

MySQL

laravel

Огляд існуючих рішень:

BOOKMOOCH

Переваги:

- сайт безкоштовний;
- цікаві зображення, концепція веб-сайту;
- приємне поєднання кольорів на сайті;
- зручна навігація веб-продукту.

Недоліки:

- проблеми з адаптивністю сайту;
- пошук книг здійснюється лише при введенні повної назви книги .

Огляд існуючих рішень:

PAPERBACKSWAP

Переваги:

- великий каталог книг;
- реалізований зручний пошук книг.

Недоліки:

- велика кількість реклами;
- надто простий дизайн сайту в плані вигляду;
- важко орієнтуватися на сайті.

Діаграма варіантів використання

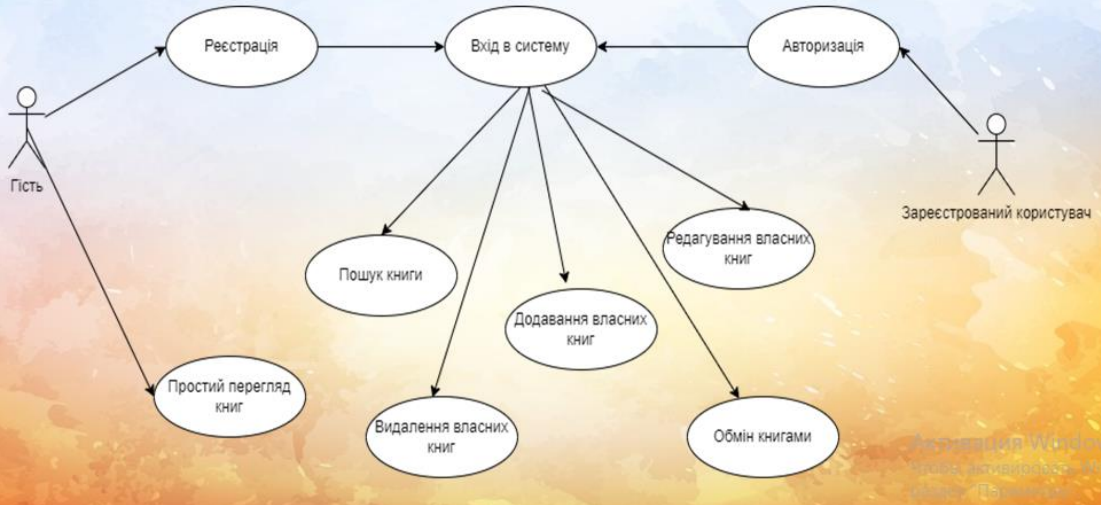
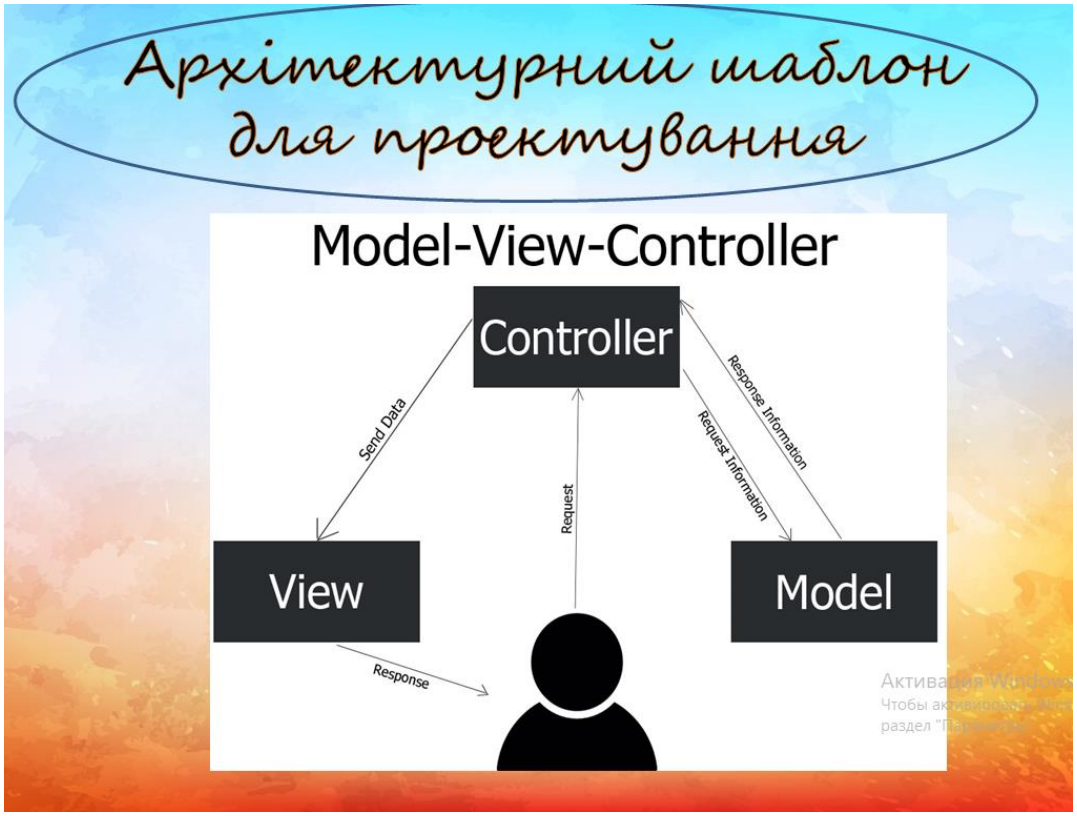
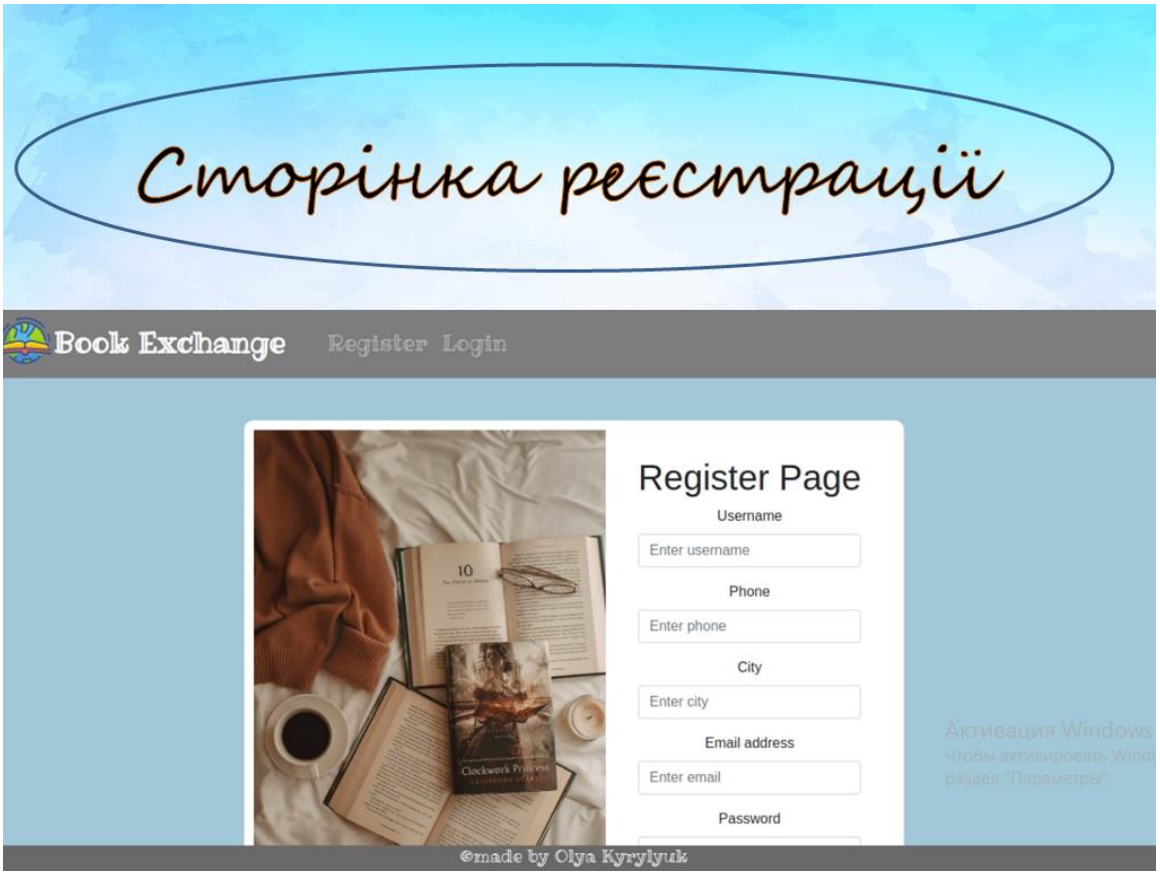


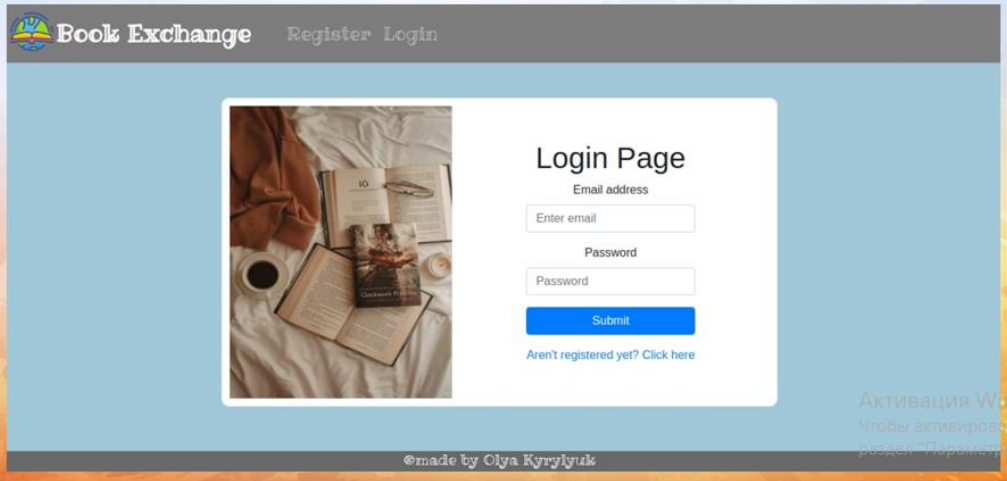

Схема бази даних
User
Id
Phone Number
Email Address
City
Password
Slug
Username
Exchange
Id
User1_id
User2_id
Book1_id
Book2_id
Status
Unread
Created_at
Updated_at
Book
Id
Name
Slug
Author
Photo
Created_at
Updated_at
Genre_id
User_id
Genre
Id
Name
Created_at
Updated_at


Архітектурний шаблон для проектування
Model-View-Controller
Controller
Send Data
Response Information
Request Information
Request
View
Model
Response


Гостьова сторінка
Book Exchange
Register Login
STRANGE LABOUR
ROBERT G. PENNER
"With this brilliant debut, Penner thoughtfully upends the tropes of postapocalyptic fiction." -- Publishers Weekly
Books
BESTSELLER
STEPHEN KING
ALICE IN WONDERLAND
Активация Windows
Чтобы активировать Wind
раздел "Параметры".


Сторінка реєстрації
Book Exchange
Register Login
Register Page
Username
Enter username
Phone
Enter phone
City
Enter city
Email address
Enter email
Password
©made by Olya Kyrylyuk
Активация Windows
Чтобы активировать Wind
раздел "Параметры".

Сторінка авторизації

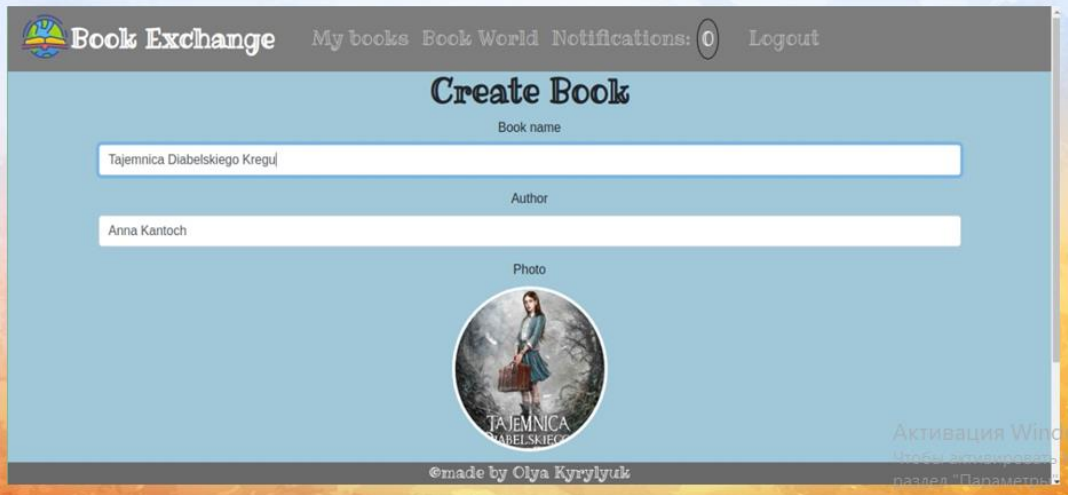


Сторінка з власними книгами(1 частина)

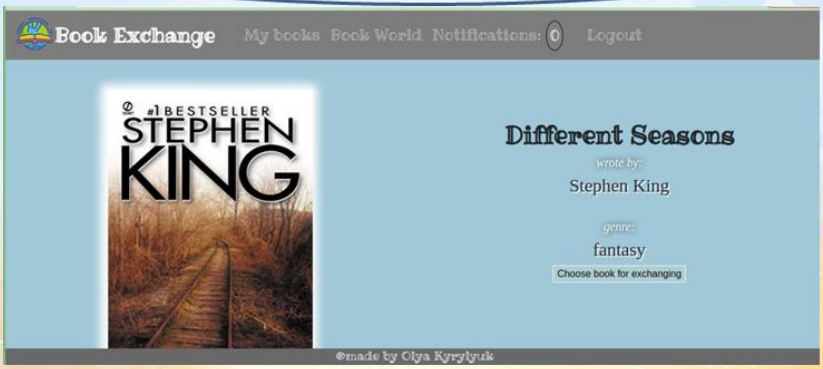
Сторінка з власними книгами(2 частина)

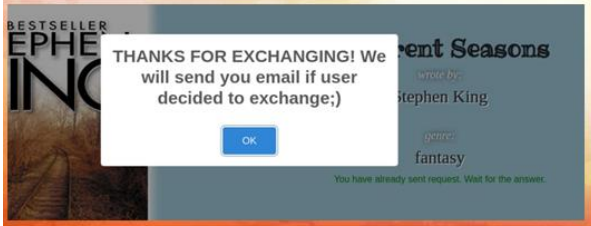
Сторінка перегляду всіх книг

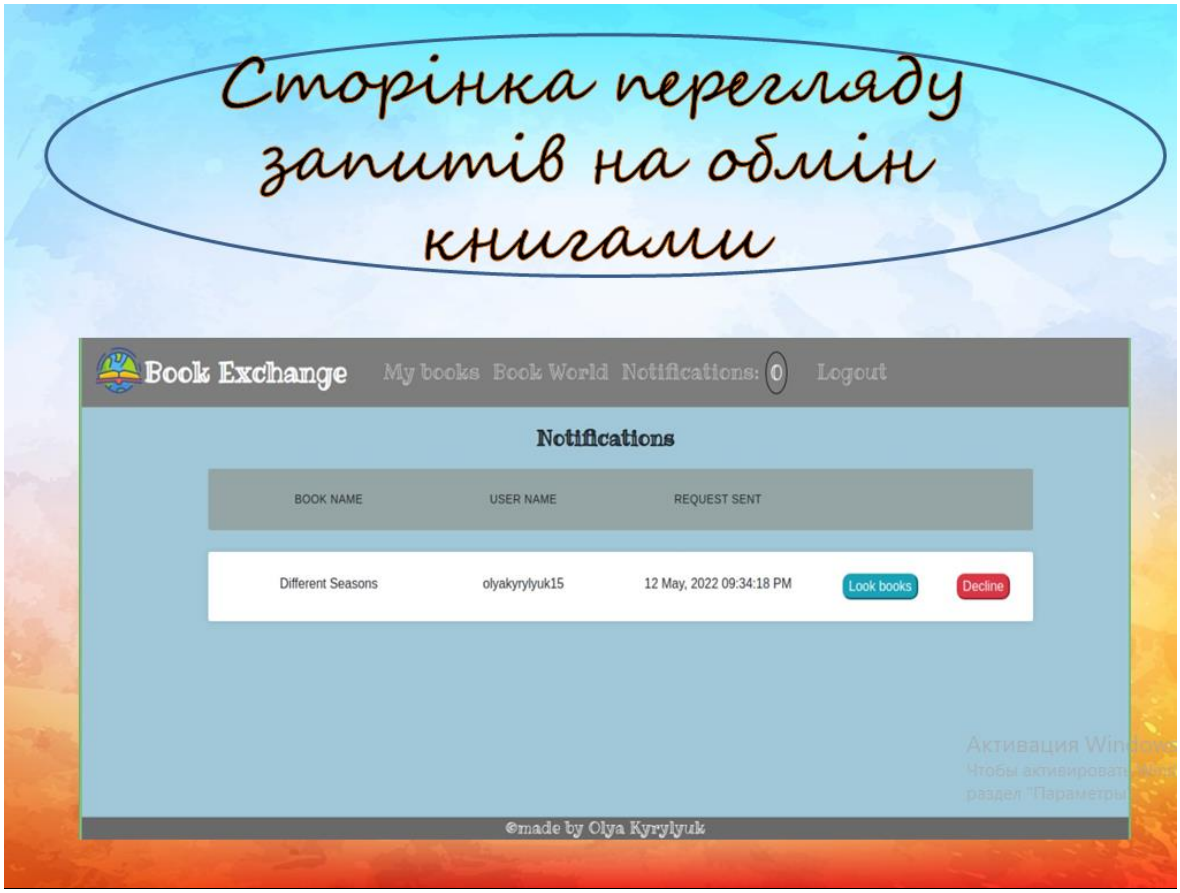
Сторінка додання однієї книги

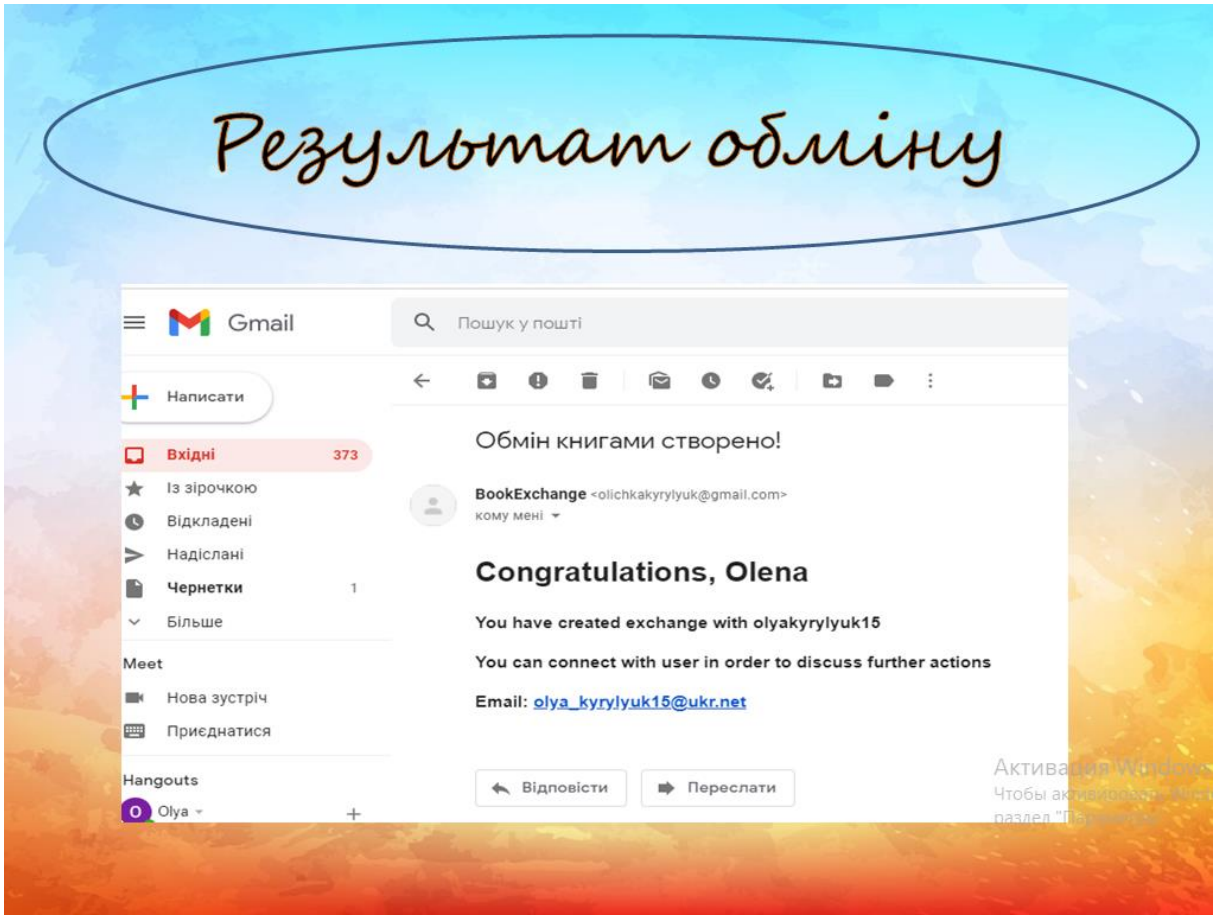


Процес обміну книгами






Сторінка перегляду запитів на обмін книгами
Book Exchange
My books
Book World
Notifications: 0
Logout
Notifications
BOOK NAME
USER NAME
REQUEST SENT
Different Seasons
olyakyrylyuk15
12 May, 2022 09:34:18 PM
Look books
Decline
©made by Olya Kyrylyuk


Результат обміну
Gmail
Пошук у пошті
Написати
Вхідні 373
Із зірочкою
Відкладені
Надіслані
Чернетки 1
Більше
Meet
Нова зустріч
Приєднатися
Hangouts
Olya
Обмін книгами створено!
BookExchange <olichkakyrylyuk@gmail.com>
кому мені
Congratulations, Olena
You have created exchange with olyakyrylyuk15
You can connect with user in order to discuss further actions
Email: olya_kyrylyuk15@ukr.net
Відповісти
Переслати


Висновок
В процесі розробки програмного забезпечення для дипломного проекту на тему: «Веб-додаток «Book Exchange» для пошуку книг та обміну ними» було досліджено предметну область, визначено вимоги, відповідно до яких було спроектовано структури проекту та бази даних. Готовий проект відповідає поставленому завданню та усім визначеним вимогам.

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Кирилюк О. О.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

01 червня 2022 р.
дата

підпис

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-19-1
Кирилюк О.О.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему дипломного проекту освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»:
Веб-додаток «Book Exchange» для пошуку книг та обміну ними

(керівник дипломного проекту – Праворська Наталія Іванівна)
Прізвище, ім'я, по батькові

01 червня 2022р.
Дата

Підпис студента

Wed May 25 20:08:22 EEST 2022, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 5.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

ID: 103992 Назва: Веб-додаток для пошуку та обміну книгами «BookExchange» Додано в БД: 2022-05-25 Автора: О.О.Кирилюк Керівники: Н. І. Праворська Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	44257	512	7825 (18%)	115 (22%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1011339349

Дата перевірки:
25.05.2022 21:05:29 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2022 21:08:42 EEST

ID користувача:
100005589

Назва документа: **Кирилюк ІПЗс-19-1 Диплом без додатків**

Кількість сторінок: **62** Кількість слів: **7536** Кількість символів: **62659** Розмір файлу: **7.64 MB** ID файлу: **1011225099**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

14.3%
Схожість

Найбільша схожість: **7.19%** з джерелом з Бібліотеки (ID файлу: **1008215667**)

5.45% Джерела з Інтернету	291	Сторінка 64
9.7% Джерела з Бібліотеки	114	Сторінка 65

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	25
Підозріле форматування	25 сторінок

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Кирилюк Ольга Олександрівна

Тема Веб-додаток «Book Exchange» для пошуку книг та обміну ними

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень 0; кількість сторінок записки 103
1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті було досліджено предметну область, проаналізовано існуючі подібні рішеннч, визначено фенкціональні та нефункціональні вимоги, спроектована структура веб-додатку а також структура бази даних, створені UML-діаграми, розроблене програмне забезпечення та протестовано проект на коректність роботи.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект виконаний відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі було визначено мету та завдання дипломного проектування, доведено актуальність теми. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. В четвертому розділі було виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми.

4. Позитивні сторони проекту Дана тема є досить актуальною, оскільки користувач може обмінювати книгами з іншими користувачами, що дозволяє зекономити кошти.

5. Негативні сторони проекту У проекті пошук книг був реалізований лише за назвою книги – було б доцільно додати пошук за автором.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконано відповідно до теми дипломного проекту та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про дипломний проект в цілому Дипломний проект заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики дипломного проекту. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект виконаний у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «відмінно».

РЕЦЕНЗЕНТ Говорущенко Тетяна Олександрівна, доктор технічних наук, завідувач кафедри комп'ютерної інженерії та інформаційних систем.

"01" червня 2022 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Веб-додаток «Book Exchange» для пошуку книг та обміну ними»

Автор: Кирилюк Ольга Олександрівна
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Праворська Наталія Іванівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 14,3 % і адресується до 405 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ Н.І. Праворська

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк