

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБМІНУ КНИГАМИ

(назва теми)

BOOK SHARING SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-220сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ПІДГОРНА Ю.Ю.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТІМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

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

ПІДГОРНОЇ Юлії Юріївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення обміну книгами. Book Sharing Software

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

_____ Юлія ПІДГОРНА
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 94 с., 26 рис., 3 табл., 3 дод., 13 джерел.

КНИГА, ОБМІН КНИГАМИ, ПИСЬМЕННИК, ЗАСТАВА, ВЕБФРЕЙМВОРК.

Об'єкт роботи – процес обміну книгами. Предмет роботи – програмне забезпечення обміну книгами. Мета роботи – розробити програмне забезпечення обміну книгами для підтримки можливостей продовження спільного використання книг після їхнього першого придбання.

Матеріали, методи та технічні засоби: мова програмування Python, засоби веброзробки, включаючи HTML, CSS, вебфреймворк Django, та система MySQL для роботи з базою даних.

Результати. Запроєктоване на основі результатів аналізу предметної області, з врахуванням основних виділених вимог і особливостей програми, реалізоване і відповідним чином описане програмне забезпечення у вигляді вебдодатку є результатом цієї роботи.

Висновки. Створена програма надає можливість не тільки передати тимчасово книгу в користування іншій людині, але і фактично підтримувати розширення її подальшого використання за рахунок страхування інтересів власника на основі внесення застави, за рахунок надання гнучкої системи варіантів обміну та надання інформації про орендарів і конкретні екземпляри книг, а також конкретні видання книг на основі використання рецензування і відгуків.

Галузь використання. Люди, які мають друковані книги у власній бібліотеці й готові надавати їх у тимчасове використання за внесення застави й за інших умов, та люди, які знаходяться в пошуку книг, зокрема достатньо рідкісних.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 pages, 26 figures, 3 tables, 3 appendixes, 13 sources.

BOOK, BOOK EXCHANGE, WRITER, PLEDGE, WEBFRAMEWORK.

The object of the work is the book sharing process. The subject of work is book sharing software. The goal of the work is to develop book sharing software to support continued sharing of books after their initial purchase.

Materials, methods and technical means: Python programming language, web development tools including HTML, CSS, Django web framework, and MySQL database system.

The results. The software in the form of a web application, designed on the basis of the results of the analysis of the subject area, taking into account the main selected requirements and features of the program, implemented and described accordingly, is the result of this work.

Conclusions. The created program provides an opportunity not only to temporarily transfer a book for use by another person, but also to actually support the expansion of its further use by insuring the interests of owners on the basis of a deposit, by providing a flexible system of sharing options and providing information about tenants and specific copies of books, and as well as specific editions of books based on the use of reviews and feedback.

Field of use. People who have printed books in their own library and are willing to provide them for temporary use with a deposit and other conditions, and people who are in search of books, in particular sufficiently rare ones.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Огляд програмного забезпечення для обміну книгами.....	9
1.2 Порівняння аналогів	12
1.3 Висновки за розділом 1	13
2 Проєктування програми.....	14
2.1 Вибір засобів проєктування і розробки	14
2.2 Визначення функціональних вимог до програмного забезпечення	15
2.3 Проєктування структури бази даних.....	18
2.4 Висновки за розділом 2	32
3 Розробка програми	34
3.1 Порядок функціонування програми.....	34
3.2 Реалізація функцій додатку.....	36
3.3 Висновки за розділом 3	45
4 Експлуатація та тестування програми	46
4.1 Опис програми.....	46
4.2 Експлуатація програми.....	46
4.3 Тестування програми.....	51
4.4 Висновки за розділом 4	53
Висновки	54
Перелік джерел посилання	55
Додаток А Технічне завдання	57
Додаток Б Текст програми	62
Додаток В Слайди презентації.....	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

За даними дослідження [1] глобальні продажі книг, які були у використанні, зростають швидше, ніж продажі нових книг, при чому ця тенденція за очікуваннями має продовжуватися протягом наступного десятиліття:

- загальні цифри станом на 2022 рік відповідають обсягу ринку книг, які були у використанні, у 24,03 мільярди доларів, зі зростанням з 2021 року на 5,5%;

- очікування щодо наступних 10 років передбачають, що обсяг ринку зросте майже в 2 рази (до понад 45 мільярдів доларів);

- на 2020 рік ринок книг, які були у використанні, займає 15 % ринку книг загалом у світі;

- більше 50% усіх продажів книг, які були у використанні, стосуються США та Європи [1]-[3].

Книги, які були у використанні, часто є вибором людей, коли вони обмежені в ресурсах, а процеси, в яких вони задіяні, допускають можливість використання не найновіших видань. Наприклад, студенти можуть вибирати в якості навчальних матеріалів книги, які були у використанні. Окрім того певні книги можуть бути не доступні в книжкових магазинах у нових виданнях, тоді ринок вживаних книг є джерелом для отримання унікальних видань [4].

Тож загалом варто вважати, що ринок вживаних книг є таким, що розвивається, а певні люди в певні періоди стикаються з ним як з одним з небагатьох шляхів вирішення власних проблем. Тому розробка програмного забезпечення (ПЗ) для підтримки процесів на цьому ринку є важливою. Мета роботи – розробити ПЗ обміну книгами для підтримки можливостей продовження спільного використання книг після їхнього першого придбання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмного забезпечення для обміну книгами

Існує достатньо багато програм, які стосуються книжок, проте щодо підтримки обміну книги коло дещо звужується. У цьому огляді необхідно було не тільки знайти такі програми, але і виділити їх можливості, щоб зрозуміти, яким чином їх можна інтегрувати для вдалого застосування у складі ПЗ обміну книгами.

Для цієї сфери важливим засобом є засіб, який надає інформацію про те, на скільки книги є добрими. Тобто коли ми шукаємо книгу за деякою темою, нам важливе сприйняття людей, які ці книги вже прочитали. Тому першим виокремлено вебсайт GoodReads [5] (рис 1.1).

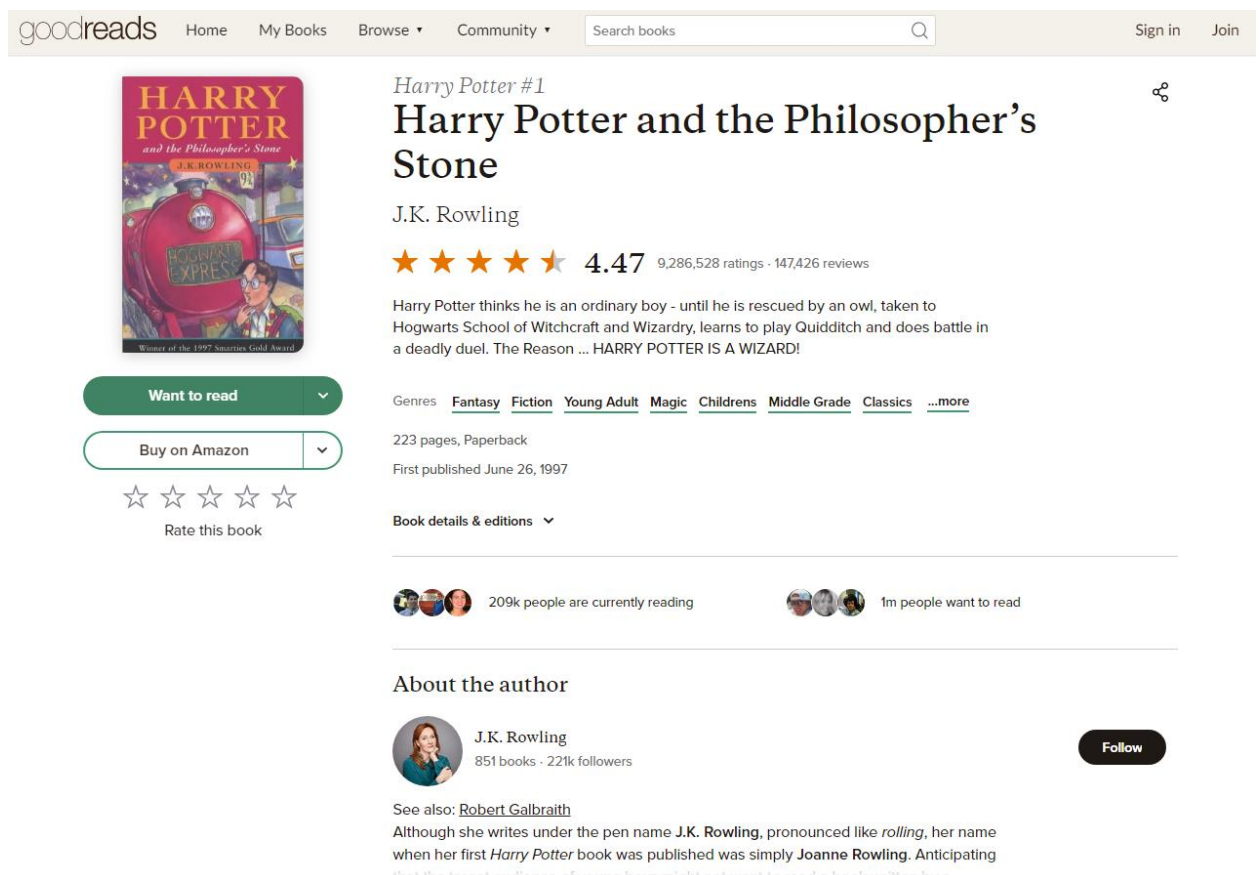


Рисунок 1.1 – Сторінка книги в GoodReads

GoodReads (рис 1.1) – вебсайт, який дозволяє отримувати загальну

інформацію про книжки, ділитися власними враженнями, бачити враження інших людей, фіксувати книжки, які прочитав користувач. Люди можуть відзначати, які книжки вони читають в даний момент, створювати власні рецензії книжок, при чому з часом створювати нові рецензії про ті самі книжки. Це достатньо зручний вебсайт для того, щоб фіксувати книжки, які прочитав користувач (призначення для самого користувача), та щоб знаходити книжки для читання в подальшому (призначення для інших користувачів).

Але важливим є те, що рецензії створюються щодо книжок загалом, а не конкретних видань. Тому фактично це оцінка тексту, створеного письменником, а не фізичного видання.

Щодо обміну книжок загальновідомим зараз у світі є буккросинг. В якості одного з прикладів виокремлено Little Free Library [6] (рис. 1.2).

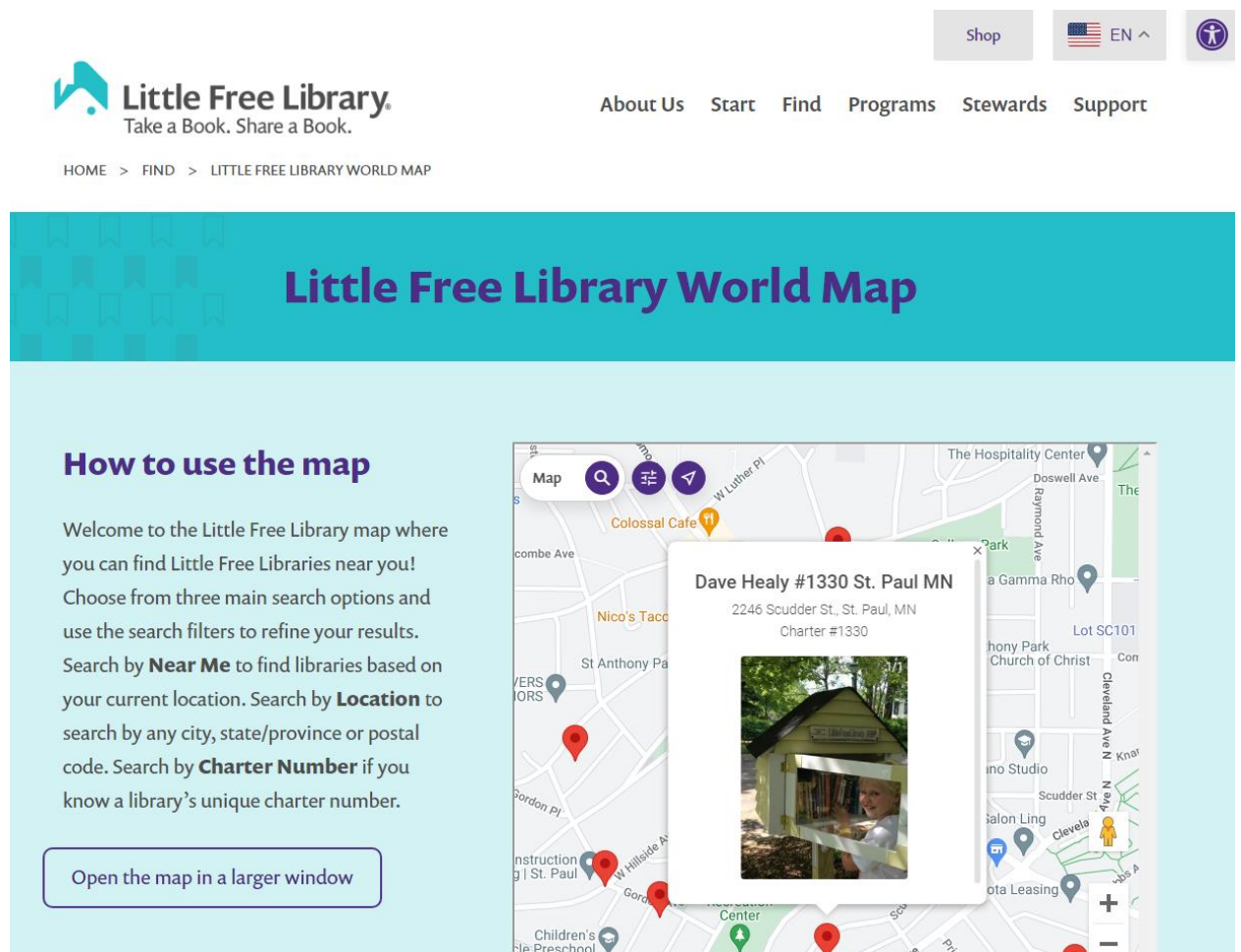


Рисунок 1.2 – Сторінка пошуку бібліотек у Little Free Library

Потрібно вказати на те, що таких рішень багато. Фактично Little Free Library об'єднує людей, які готові створити невелику бібліотеку біля свого будинку, де будуть залишати власні книги інші люди або брати книги, які залишені іншими людьми. Такі бібліотеки звичайно існують не тільки на цьому сайті, але і фізично в багатьох містах світу. Проте у такому випадку власник книги фактично відмовляється від власництва. Книга продовжує жити, але наприклад, стан її може псуватися. Тому це не всіх власників книжок фактично задовольняє в повній мірі.

Share A Book [7]-[8] (рис. 1.3) представлений в соціальній мережі Facebook, хоча на даний момент розробляється вебсайт під це рішення.



Рисунок 1.3 – Продаж книги, що була у використанні, на Share A Book

Share A Book дозволяє продавати книги, які вже були у використанні. Звичайно, що це не єдине подібне рішення, і фактично на будь-якому вебсайті, що надає дошку оголошень, можуть зустрічатися оголошення про продаж

книжок, які були у використанні. Щодо кожної книжки є опис стану. Проте виконати пошук за специфічними критеріями не можна. На вебсайті, що створюється, планується підтримка можливості просто віддати книжку іншій людині, при чому заявлено, що це саме навчальні матеріали. Тобто фактично всі ці варіанти також, як і в буккросингу, пов'язані з тим, що право власництва припиняється.

1.2 Порівняння аналогів

У процесі огляду було виділено важливі концептуальні функції кожного вебсайту, який було винесено в підсумковий підрозділ 1.1, хоча було проаналізовано ширше коло рішень. Проведений аналіз встановив, що як такого готового рішення для підтримки саме обміну книжками з забезпеченням ідеї внесення залогу немає. Цю ідею вирішено було реалізувати в дипломній кваліфікаційній роботі (табл. 1.1).

Таблиця 1.1 – Порівняння аналогів з розробкою

Характеристика	GoodReads	Little Free Library	Share A Book	Розробка
Рецензії	Книги	—	Коментарі в Facebook	Видання, екземпляри книг
Обмін книгами	—	Без сплати	Планується без сплати	Зі сплатою, без сплати, з заставою
Замовлення книг для обміну	—	—	—	Так
Придбання книг, що були у використанні	—	—	Так	Так

1.3 Висновки за розділом 1

Досягнення мети можливо при успішному виконанні наступних завдань роботи:

- огляд і аналіз ПЗ, що може використовуватися для обміну книгами;
- проєктування ПЗ обміну книгами;
- реалізація ПЗ обміну книгами і тестування.

Проведено огляд і аналіз ПЗ, що може використовуватися для обміну книгами, включав зокрема GoodReads, Little Free Library та Share A Book, що дозволило визначити концепцію ПЗ, що розробляється в роботі, включаючи можливості рецензування, але вже не книг загалом як в GoodReads, а видань і екземплярів, та обміну книгами на основі як надання у тимчасове використання з можливістю внесення залогу, так і надання у подальше безкоштовне поширення, обміну книгами і придбання за потреби, що не є основним призначенням ПЗ.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Python є однією з мов програмування, які підтримують веброзробку, при цьому надаючи велику кількість додаткових, добре задокументованих засобів [9]. Мова також має багато вебфреймворків, що як раз і робить її популярною у веброзробці. У роботі було порівняно Django [10] і Pyramid [11]-[12] (табл. 2.1). Django обрано для розробки, зважаючи на більшу кількість включених одразу інструментів та можливість розробки на основі окремих проєктів, що дозволяє інтегрувати, наприклад, авторизацію і реєстрацію на основі записів Google замість створення власного рішення, при цьому спрощуючи цю інтеграцію.

Таблиця 2.1 – Порівняння вебфреймворків для розробки ПЗ обміну книгами

Показник	Django	Pyramid
Підтримка Bootstrap	+	+
Архітектурний патерн	Model-View-Controller	Model-View-Controller
Система підтримки шаблонів	+	Можливо підключити
Робота зі структурою додатку	Можливість створювати декілька додатків у одному проєкті	Один проєкт як один додаток
Інтерфейс адміністратора	+	—
Підтримка високого трафіку на практиці	+	+
Підтримка щодо документації	++	+

2.2 Визначення функціональних вимог до програмного забезпечення

Для забезпечення роботи користувачів, які є власниками книг і/або які знаходяться в пошуку книг, які вже були у використанні, було вирішено передбачити сценарії, які надаються кожному користувачу, при цьому забезпечуючи ідентифікацію облікового запису всюди, де це необхідно для отримання відповідної інформації. Тому весь набір сценаріїв не був розподілений на різні ролі користувачів, що має дозволити в підсумку працювати всім користувачам з одним і тим самим інтерфейсом і розуміти всю широту можливостей, що надається програмою. Загальна множина сценаріїв включає:

- пошук екземплярів книжок для обміну;
- пошук книжок;
- пропонування книжок за зацікавленостями користувача;
- внесення власного екземпляру книжки для обміну;
- керування внесеними екземплярами книжок користувача;
- внесення екземпляру відсутньої в системі книги;
- відтворення списку замовлень за екземпляром книги;
- відтворення списку екземплярів видання книги;
- замовлення тимчасової оренди книг;
- замовлення подальше поширення книг;
- замовлення обміну книг;
- керування станом замовлення;
- відтворення списку створених замовлень користувачем;
- відтворення списку замовлень щодо книг користувача;
- придбання книги;
- внесення власної інформації користувача;
- реєстрація користувача;
- керування власною інформацією користувача;

- керування адресами користувача;
- відтворення інформації про письменника;
- відтворення інформації про користувача;
- відтворення списку книжок письменника;
- відтворення рейтингу видань книжок за жанром;
- відтворення рейтингу видань книжок за тегом;
- відтворення видань книги;
- відтворення всіх рецензій за книгою;
- відтворення всіх прорецензованих видань користувачем;
- відтворення всіх відгуків користувача;
- відтворення всіх відгуків про користувача;
- відтворення всіх відгуків про екземпляр книги;
- відтворення інформації про книгу;
- відтворення інформації видання книги;
- відтворення інформації про екземпляр книги;
- створення заявки на власництво екземпляру книги;
- створення співбесіди з користувачем;
- надсилання повідомлення у співбесіді;
- відтворення списку співбесід користувача;
- внесення правила оренди книг;
- керування внесеними правилами оренди книг;
- рецензування видання книги;
- внесення відгуку про екземпляр книги;
- внесення відгуку про користувача;
- відтворення списку письменників за країною;
- відтворення списку письменників за містом;
- відзначення зацікавленості користувача у виданні;
- відзначення зацікавленості користувача у книгах письменника;
- відтворення книг, виданих видавництвом (рис. 2.1).

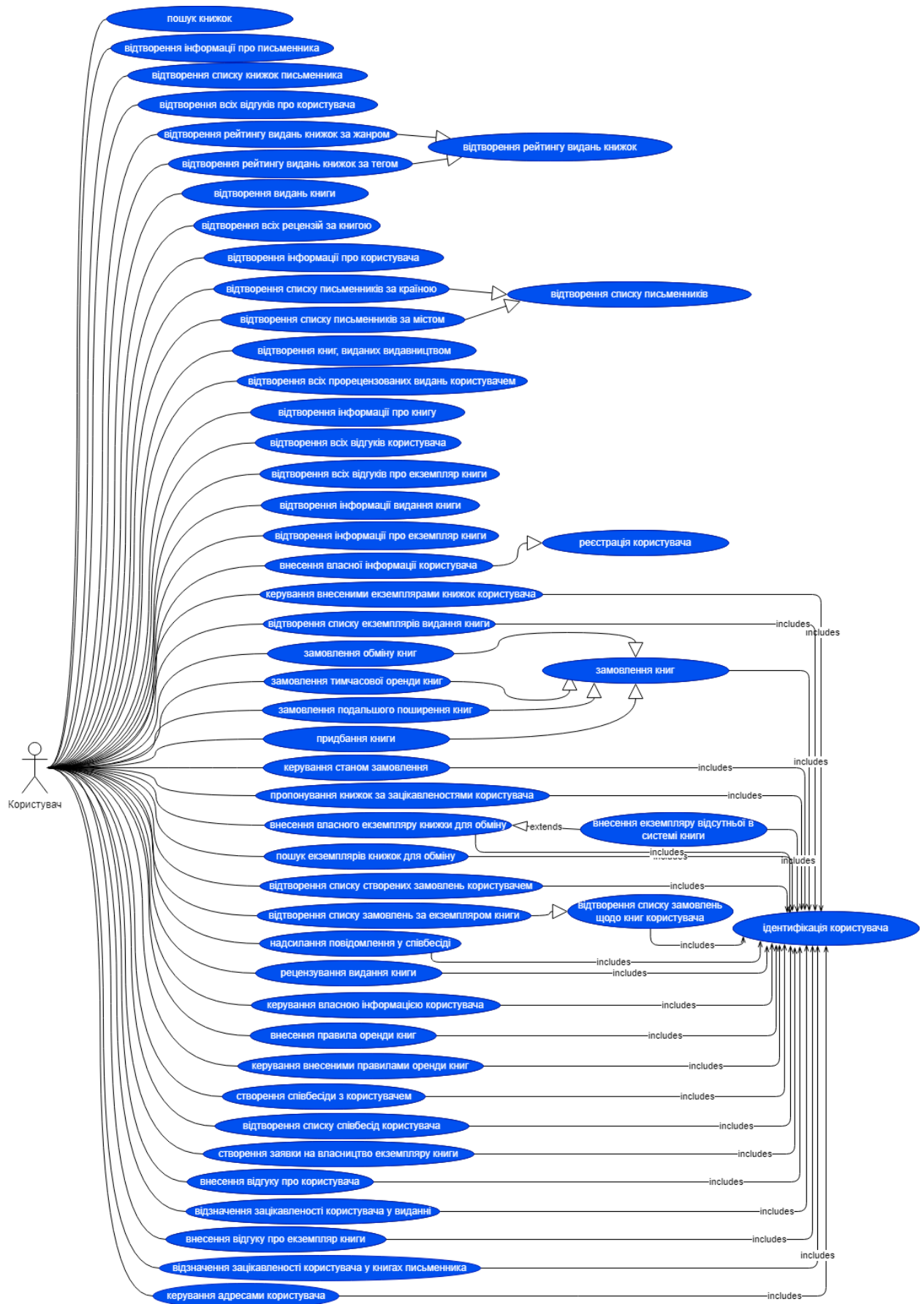


Рисунок 2.1 – Діаграма прецедентів

2.3 Проектування структури бази даних

Враховуючи сценарії роботи з програмою, для забезпечення основи роботи необхідна база даних (БД), яка дозволить зберігати дані між різними сеансами роботи різних користувачів. Підтримка роботи БД виконується через систему MySQL, що при розгортанні вебдодатків різного рівня використовується різними організаціями, таким чином підтверджуючи власну придатність для розв'язання таких проблем.

Для будь-якої книги її автор є однією з основних характеристик, тому сутність письменника було внесено для зручності внесення даних у БД у вигляді таблиці Writer з полями:

- id – первинний ключ у таблиці, ідентифікатор письменника;
- full_name – прізвище та ім'я письменника;
- born_city – місто, в якому народився письменник: визначається зовнішнім ключем з таблиці City;
- born_date – дата народження письменника;
- end_date – дата смерті письменника (відповідно для живих письменників залишається порожньою);
- living_country – країна, в якій живе або громадянство якої має письменник: визначається зовнішнім ключем з таблиці Counter;
- about_text – інформація про письменника;
- photo – фотографія письменника.

Для класифікації книг одним з традиційних засобів є жанри, тому створена таблиця Genre, що містить назву жанра у полі name та ідентифікатор у полі id.

Оскільки обмін книгами здійснюється у друкованому вигляді, то однією з важливих характеристик книг є видавець, який публікує книги. Інформація про видавців винесена в таблицю Publisher з полями:

- id – первинний ключ у таблиці, ідентифікатор видавця;
- name – назва видавця;

– `headquarters_city` – місто, де знаходиться головний офіс видавця: визначається зовнішнім ключем з таблиці `City`.

Книги видаються певними мовами, тому наступною стандартною характеристикою книги, яку через це було винесено в окрему таблицю є `Language`. Таблиця містить назву мови у полі `name` та ідентифікатор у полі `id`.

Зовнішніми ключами уже було представлено дві стандартні таблиці. Це таблиця з назвами країн `Country`, що містить назву країни у полі `name` та ідентифікатор у полі `id`, та таблиця `City`, що містить назву міста у полі `name`, ідентифікатор у полі `id` та країну, в якій є це місто, в полі `country` з зовнішнім ключем.

Безпосередньо книга є окремою сутністю і в БД базова інформація про книги знаходиться в таблиці `Book` з полями:

- `id` – первинний ключ у таблиці, ідентифікатор книги;
- `short_title` – коротка назва книги;
- `full_title` – повна назва книги;
- `photo` – фотографія книги;
- `first_published` – рік, коли було вперше опубліковано книгу;
- `about_text` – інформація про книгу;
- `approved` – підтвердження відображення цієї книги: необхідно для книг, які відсутні у системі, але які вносять звичайні користувачі, щоб їх можна було спочатку перевірити, а вже потім відображати.

У книги може бути не один автор, а декілька. При тому кожен письменник може бути автором декількох книг. Тому даний зв'язок створено через зв'язуючу таблицю `BookWriter`, яка відповідно має 2 поля: `book` з номером книги (зовнішній ключ з таблиці `Book`), `writer` з номером письменника-автора книги (зовнішній ключ з таблиці `Writer`).

Книги, як було вказано вище, характеризуються жанрами. Кожен жанр має різні книги, проте кожна книга може мати різні жанри. Реалізовано цей зв'язок через таблицю `BookGenre`, яка відповідно має 2 поля: `book` з номером

книги (зовнішній ключ з таблиці Book), genre з номером жанру (зовнішній ключ з таблиці Genre).

Одна й та сама книга може бути опублікована різними видавництвами. Інформація про кожну видану книгу внесена в БД у таблицю Publishedbook з полями:

- id – первинний ключ у таблиці, ідентифікатор виданої книги;
- book – книга, яку було видано: визначається зовнішнім ключем з таблиці Book;
- book_type – тип видання книги: з м'якою обкладинкою, з твердою обкладинкою, подарункове;
- paper_type – тип паперу, на якому видано книгу;
- paper_size – формат книги (розмір паперу);
- photo – фотографія книги;
- publisher – видавець книги: визначається зовнішнім ключем з таблиці Publisher;
- average_critique – усереднена оцінка книги за рецензіями користувачів;
- first_published_year – рік, у який вперше було зроблено це видання;
- illustrations – наявність ілюстрацій у книзі;
- pages_number – кількість сторінок;
- approved – підтвердження відображення цієї книги: необхідно для книг, які відсутні у системі, але які вносять звичайні користувачі, щоб їх можна було спочатку перевірити, а вже потім відображати.

Як було зазначено вище, книги публікуються певними мовами. Тобто фактично мова характеризує конкретне видання книги. Можлива ситуація, коли в книзі якась мова є основною, але зустрічаються і інші мовами. Для цього з врахуванням такого типу зв'язку було створено таблицю PublishedbookLanguage, яка відповідно має 2 поля: publishedbook з номером видання книги (зовнішній ключ з таблиці Publishedbook), language з номером мови (зовнішній ключ з таблиці Language).

Видання книги має різні екземпляри, якими володіють різні люди, які можуть захотіти обмінятися книгами. Інформація про екземпляр книги внесена в БД у таблицю Bookinstance з полями:

- id – первинний ключ у таблиці, ідентифікатор екземпляру книги;
- publishedbook – видання книги, яке представляє цей екземпляр книги: визначається зовнішнім ключем з таблиці Publishedbook;
- owner_account – користувач, який володіє на даний момент книгою: визначається зовнішнім ключем з таблиці Account;
- condition – числове значення, яке представляє стан, в якому знаходиться цей екземпляр книги;
- condition_statement – детальний опис стану, в якому знаходиться цей екземпляр книги;
- longcode – довгий код, який може бути згенерований для книги, якщо власник згоден фізично розмістити його на сторінці книги, щоб її унікально ідентифікувати і в подальшому відслідковувати її долю, виходячи з отриманого екземпляру, адже книга може бути в якийсь момент розповсюджена і не через систему, а людина, яка її отримала, може вирішити внести цю інформацію в систему;
- published_year – рік публікації екземпляру книги: він може відрізнятися від року, вказаного для видання книги, адже видання може в подальшому друкуватися повторно;
- photo – фотографія екземпляру книги;
- average_review – усереднена оцінка з відгуків користувачів, які брали її;
- buy_option – можливість придбати цей екземпляр книги, якщо таке бажання виникне у людини, яка отримала цю книгу на час: визначається нульовим значенням, коли книгу придбати неможливо, або вартістю такого придбання;
- deposit_amount – застава, яку має бути внесено за обмін, тобто сума, яка повертається замовнику;

– `shared_option` – доступний варіант обміну для книги: отримати на час, забрати повністю для подальшого поширення, обміняти книги або обміняти чи отримати на час;

– `shared` – доступність книги для обміну в поточний момент;

– `delivered` – доступність книги для доставки, тобто чи вона може бути відправлена службою доставки, чи її можна отримати тільки у власному місті;

– `approved` – підтвердження відображення цієї книги.

Оскільки екземпляр книги може розповсюджуватися не тільки через цю програму, то в певний момент може виявитися, що книга вже змінила власника. Для встановлення цього факту та внесення відповідних даних створено окрему сутність заявки на володіння. Оскільки книга може мати довгий код, який її ідентифікує, то це дозволяє зрозуміти користувачу, що інформація про цей екземпляр книги вже була внесена раніше. Внесення цієї інформації відбувається в таблицю `BookinstanceApplication` з полями:

– `id` – первинний ключ у таблиці, ідентифікатор заявки на володіння екземпляром книги;

– `bookinstance` – екземпляр книги, якого стосується заявка: визначається зовнішнім ключем з таблиці `Bookinstance`;

– `owner_account` – користувач, який подав заявку: визначається зовнішнім ключем з таблиці `Account`;

– `when_created` – час створення заявки;

– `approved` – підтверджена або скасована заявка;

– `when_approved` – час підтвердження заявки;

– `photo` – фотографія на підтвердження власництва.

Обміни книгами можуть відбуватися на основі різних процедур. Першим варіантом процедури є отримання книг від їхнього власника на час. Воно може здійснюватися за певну сплату, з внесенням застави або безкоштовно. Інформація про такий обмін зберігається в таблиці `Shareorder` з полями:

- id – первинний ключ у таблиці, ідентифікатор обміну (фактично оренди) книг;
- account – користувач, який оформив це замовлення: визначається зовнішнім ключем з таблиці Account;
- when_created – час створення замовлення обміну (оренди) книг;
- stage – поточний стан замовлення обміну (оренди) книг;
- when_approved – час підтвердження замовлення власником книг: видача згоди на те, щоб передати книги;
- when_received – час фактичного отримання книг користувачем (замовником) від власника;
- to_return – час, визначений власником, коли користувач повинен повернути книги;
- when_returned – час, коли було повернуто книги власнику;
- pay_amount – сплата, яку має бути внесено за сам обмін (оренду), тобто сума, яка не повертається замовнику;
- deposit_amount – застава, яку має бути внесено за обмін, тобто сума, яка повертається замовнику;
- bank_amount – сума, яку було внесено на банківський рахунок за цим замовленням на даний момент.

Обмін книг може включати не тільки одну книгу, а декілька. Тому склад замовлення потрібно було винести окремо – у таблицю ShareorderBookinstances БД з полями:

- shareorder – замовлення з обміну (оренди) книг, до якого входить цей вміст: визначається зовнішнім ключем з таблиці Shareorder;
- bookinstance – екземпляр книги, який входить у замовлення з обміну (оренди) книг: визначається зовнішнім ключем з таблиці Bookinstance.

Другим варіантом процедури обміну книгами є безпосередньо обмін, коли один власник надає одну або декілька книг, а інший надає йому на обмін одну або декілька своїх. Кількість книг з кожної сторони не регламентована, а вирішується індивідуально: хто на що готовий погодитися. При цьому маєтсья

на увазі обмін, коли книги потім потрібно повернути. Якщо книги не повертаються, то використовується подвійне замовлення, коли кожен замовляє у іншої сторони певний набір книг у варіанті без повернення, який описаний нижче.

Тож тимчасовий обмін книгами визначається в таблиці Changeorder з полями:

- id – первинний ключ у таблиці, ідентифікатор тимчасового обміну книг;
- account – користувач, який оформив замовлення початково: визначається зовнішнім ключем з таблиці Account;
- to_account – користувач, з яким буде відбуватися обмін: визначається зовнішнім ключем з таблиці Account;
- stage – поточний стан замовлення тимчасового обміну книг;
- when_created – час створення замовлення тимчасового обміну книг;
- when_approved – час підтвердження замовлення другою стороною (поле to_account): видача згоди на те, щоб передати книги;
- when_received – час фактичного виконання обміну книг;
- to_return – час, коли за згодою сторін потрібно повернути книги;
- when_returned – час, коли було повернуто книги.

У таблиці потрібні 2 поля для збереження користувачів. Ці дані можна було б витягти зі складу відповідного замовлення, але можлива ситуація, коли замовник хоче вказати, які книги він готовий віддати, після співбесіди з іншим користувачем, а той у відповідь визначає, які він готовий надати. Тоді витягти інформацію про обидві сторони зі складу замовлення неможливо. Для таких ситуацій як раз і зафіксовано обидва поля в таблиці. Окрім того це призводить до меншої кількості дій при роботі з даними.

Вміст тимчасової оренди книг знаходиться в таблиці ChangeorderBookinstances з полями:

- id – первинний ключ у таблиці, ідентифікатор вмісту тимчасового обміну книг;

– `changeorder` – замовлення тимчасової оренди книг, якої стосується цей вміст;

– `bookinstance` – екземпляр книги, який входить до замовлення;

– `approved` – підтвердження на обмін цієї книги обома сторонами.

Третім варіантом процедури обміну книгами є подальше поширення, коли користувач отримує книгу або книги і далі не повинен її повертати, а після прочитання може передати іншому користувач тим чи іншим способом. Тобто фактично це вже варіант буккросингу. Інформація про такий варіант обміну знаходиться в таблиці `Openshareorder` з полями:

– `id` – первинний ключ у таблиці, ідентифікатор замовлення обміну (подальшого поширення) книг;

– `account` – користувач, який оформив замовлення обміну (подальшого поширення) книг: визначається зовнішнім ключем з таблиці `Account`;

– `stage` – поточний стан замовлення обміну (подальшого поширення) книг;

– `when_created` – час створення замовлення обміну (подальшого поширення) книг;

– `when_approved` – час підтвердження замовлення обміну (подальшого поширення) книг: видача згоди на те, щоб передати книги;

– `when_received` – час фактичного виконання обміну книг.

Замовлення подальшого поширення книг також може включати і одну, і декілька книг, а книга за час свого життя може входити в різні такі замовлення, тому було створено зв'язуючу таблицю `OpenshareorderBookinstance`, яка відповідно має 2 поля: `openshareorder` з номером видання книги (зовнішній ключ з таблиці `Openshareorder`), `bookinstance` з номером мови (зовнішній ключ з таблиці `Bookinstance`).

Четвертим варіантом процедури обміну книгами, який фактично є продовженням процедури обміну, є придбання книги. Тобто цей варіант передбачає, що користувач взяв у оренду якусь книгу, але прочитавши її, вона йому сподобалась, і він вирішив залишити її в своїй бібліотеці. Для цього він

звертається до власника книги з проханням її придбати, а той або дозволяє це, або ні. Інформація про такі замовлення, які не є основними в цій системі, але які можуть траплятися, міститься в таблиці Buyorder з полями:

- id – первинний ключ у таблиці, ідентифікатор замовлення придбання книги;
- account – користувач, який бажає придбати книгу: визначається зовнішнім ключем з таблиці Account;
- bookinstance – екземпляр книги, яку користувач бажає придбати;
- stage – стан замовлення придбання книги;
- when_created – час створення замовлення придбання книги;
- when_approved – час підтвердження замовлення придбання книги;
- when_received – час фактичного виконання придбання книги;
- bank_amount – сума, яку було передано на банківський рахунок за цим замовленням.

Отримання книг від власника на час підпадає під певні правила. І звичайно, що якщо користувач має декілька книг для обміну, то ці правила повторюються. Тому щоб їх не повторювати для кожної книги власника, вони винесені в таблицю Sharingrule з полями:

- id – первинний ключ у таблиці, ідентифікатор правила з обміну книги;
- account – користувач, який як власник книг використовує це правило: визначається зовнішнім ключем з таблиці Account;
- rules_statement – текст правила обміну;
- max_days – найтриваліший період, на який можна отримати книгу;
- pay_amount – сума, яку потрібно сплатити за цю оренду, тобто це сума, яка не повертається;
- deposit – необхідність внесення застави під час оренди книги, тобто це сума, яка повертається після завершення оренди.

Власник може для декількох своїх книг використовувати те саме правило оренди. З часом правила можуть змінюватися для книги. Тим більше

можуть змінюватися і власники. Тому зв'язування книг з правилами оренди виконується через таблицю BookinstanceSharingrule з полями:

- id – первинний ключ у таблиці, ідентифікатор застосування правила з обміну до книги;
- bookinstance – екземпляр книги, для якої застосовується це правило: визначається зовнішнім ключем з таблиці Bookinstance;
- sharingrule – правило оренди книги, яке застосовується до цього екземпляру книги: визначається зовнішнім ключем з таблиці Sharingrule;
- actual – актуальність застосування цього правила до оренди цього екземпляру книги на даний момент, тобто чи це застаріла інформація, чи ні.

Рецензії користувачів стосуються видання книги загалом на відміну від відгуків, які стосуються або екземпляру книги, або користувачів. Рецензії користувачів про видання книг знаходяться в таблиці Critique з полями:

- id – первинний ключ у таблиці, ідентифікатор застосування правила з обміну до книги;
- publishedbook – видання книги, про яке написано рецензію;
- account – рецензент: визначається зовнішнім ключем з таблиці Account;
- when_created – час рецензування;
- text_critique – текст рецензії;
- points_critique – кількість балів, установлена в рецензії.

Відгуки про екземпляри книг, які фізично брав у оренду або які читав користувач, знаходяться в таблиці BookinstanceReview з полями:

- id – первинний ключ у таблиці, ідентифікатор відгуку про екземпляр книги;
- bookinstance – екземпляр книги, якого стосується відгук;
- account – користувач, що створив відгук: визначається зовнішнім ключем з таблиці Account;
- condition – стан екземпляру книги у балах (на думку користувача);
- condition_statement – детальний опис стану книги;

- when_created – час створення відгуку;
- text_review – текст відгуку;
- points_review – кількість балів, установлена у відгуку;
- photo – фотографія екземпляру книги.

Відгуки про користувачів, які давали книги у оренду, знаходяться в таблиці AccountReview з полями:

- id – первинний ключ у таблиці, ідентифікатор відгуку про користувача;
- account_share – користувач, який надав книги: визначається зовнішнім ключем з таблиці Account;
- account_author – користувач, що створив відгук: визначається зовнішнім ключем з таблиці Account;
- points_review – кількість балів, установлена у відгуку;
- when_created – час створення відгуку;
- text_review – текст відгуку.

Співбесіда користувачів щодо обміну книгами зберігається в таблиці AccountQuery через поля:

- id – первинний ключ у таблиці, ідентифікатор співбесіди;
- account_from – перша сторона співбесіди: визначається зовнішнім ключем з таблиці Account;
- account_to – друга сторона співбесіди: визначається зовнішнім ключем з таблиці Account.

Повідомлення зі співбесіди зберігаються в таблиці TextQuery через поля:

- id – первинний ключ у таблиці, ідентифікатор повідомлення зі співбесіди;
- accountquery – співбесіда, до якої відноситься повідомлення: визначається зовнішнім ключем з таблиці Account;
- message – текст повідомлення;
- when_created – час створення повідомлення.

Користувачі програми, що є як власниками книг, так і замовниками (орендарями), визначаються на основі авторизації в Django і представлені в таблиці Account через поля:

- id – первинний ключ у таблиці, ідентифікатор користувача;
- username – логін;
- first_name – ім'я;
- last_name – прізвище;
- email – поштова адреса;
- password – пароль користувача;
- is_staff – стандартне поле Django, можливість доступу до адміністративного сайту;
- is_active – стандартне поле Django, активність облікового запису;
- is_superuser – стандартне поле Django, можливість доступу до всіх дозволів;
- last_login – стандартне поле Django, остання авторизація;
- date_joined – стандартне поле Django, дата реєстрації;
- photo – фотографія користувача;
- phone_number – телефонний номер;
- bank_details – деталі для виконання платежів;
- distance – відстань, на якій готовий забирати книги користувач;
- average_reader_review – усереднена оцінка відгуків про цього користувача як про читача (орендаря);
- average_owner_review – усереднена оцінка відгуків про цього користувача як про власника книг.

Книги у користувача можуть знаходитися за різними адресами, тому з одним користувачем може бути пов'язано декілька адрес, що зберігаються в таблиці Accountaddress через поля:

- id – первинний ключ у таблиці, ідентифікатор адреси користувача;
- account – користувач, якому належить ця адреса: визначається зовнішнім ключем з таблиці Account;

- address_city – місто, де розташована ця адреса: визначається зовнішнім ключем з таблиці City;
- address – повна адреса (вулиця, будинок);
- address_long – широта позиції;
- address_latd – довгота позиції;
- base – базовість адреси: це адреса, за якою виконується пошук книг для користувача, або це адреса, за якою у користувача можуть знаходитися деякі книги.

Зв'язування екземплярів книг з адресами виконується окремо. Ця інформація знаходиться в таблиці BookinstanceAddress через поля:

- id – первинний ключ у таблиці, ідентифікатор зв'язування екземпляру книги з адресою;
- bookinstance – екземпляр книги: визначається зовнішнім ключем з таблиці Account;
- accountaddress – адреса, за якою цей екземпляр книги знаходиться: визначається зовнішнім ключем з таблиці Accountaddress;
- when_added – час внесення даних про адресу.

Користувач може проактивно виявляти свою позицію, позначаючи, які книжки він шукає. Для цього він може позначати, в яких авторах він зацікавлений, а також в яких саме виданнях книжок. Це два варіанти виявлення власного зацікавлення.

Інформація про пошук книжок деякого письменника зберігається в таблиці WriterWanting через поля:

- id – первинний ключ у таблиці, ідентифікатор зацікавленості користувача у книжках певного письменника;
- writer – письменник, в книжках якого зацікавлений користувач: визначається зовнішнім ключем з таблиці Writer;
- wanting_account – користувач, який зацікавлений у книжках цього письменника: визначається зовнішнім ключем з таблиці Account;

- `max_distance` – максимальна відстань, на якій можна пропонувати книжки цього автора, тобто максимальна відстань від базової адреси користувача, за якої він готовий розглянути книжки цього автора в якості варіанта обміну;

- `delivered` – готовність до використання служб доставки для обміну: за готовності можуть пропонуватися варіанти у всій країні.

Зацікавленість користувача у певному виданні книжки визначається в таблиці `PublishedbookWanting` через поля:

- `id` – первинний ключ у таблиці, ідентифікатор зацікавленості користувача у певному виданні;

- `publishedbook` – видання книжки, у якому зацікавлений користувач;

- `wanting_account` – користувач, який зацікавлений у книжках цього видання: визначається зовнішнім ключем з таблиці `Account`;

- `max_distance` – максимальна відстань, на якій можна пропонувати книжки цього видання, тобто максимальна відстань від базової адреси користувача, за якої він готовий розглянути книжки цього видання в якості варіанта обміну;

- `delivered` – готовність до використання служб доставки для обміну: за готовності можуть пропонуватися варіанти у всій країні;

- `exactly` – готовність розглядати тільки це видання або і інші видання цієї книги.

Для додаткової структуризації пошуку нових книг використовуються теги, які зберігаються в таблиці БД `Tag` через поле ідентифікатора `id` і назви `name`.

Один тег може використовуватися в різних книжках, але і одна книжка може позначатися декількома тегами, тому зв'язок між ними реалізовано через таблицю `BookTag` з полями `book` з таблиці `Book` та `tag` з таблиці `Tag` (рис. 2.2).



Рисунок 2.2 – Схема БД

2.4 Висновки за розділом 2

Визначено структуру БД програми з виділенням сутностей книги, видання книги, екземпляра книги, замовлень на обмін, окремо замовлення на отримання книг від їхнього власника на час, безпосередньо замовлення на

обмін одних книг на інші, замовлення на подальше поширення книг і замовлення на придбання книги, та відповідні функції і сценарії роботи користувача з цими функціями.

3 РОЗРОБКА ПРОГРАМИ

3.1 Порядок функціонування програми

Усі замовлення було розбито на 4 види, як це було представлено в розділі 2. Від станів замовлень залежить порядок опрацювання і відповідно вигляд функцій, які можуть використовувати ті чи інші користувачі в кожен момент.

Стани і порядок станів однаковий у замовлення подальшого поширення книг та покупки книги, адже тоді власник книги перестає ним бути (рис. 3.1).

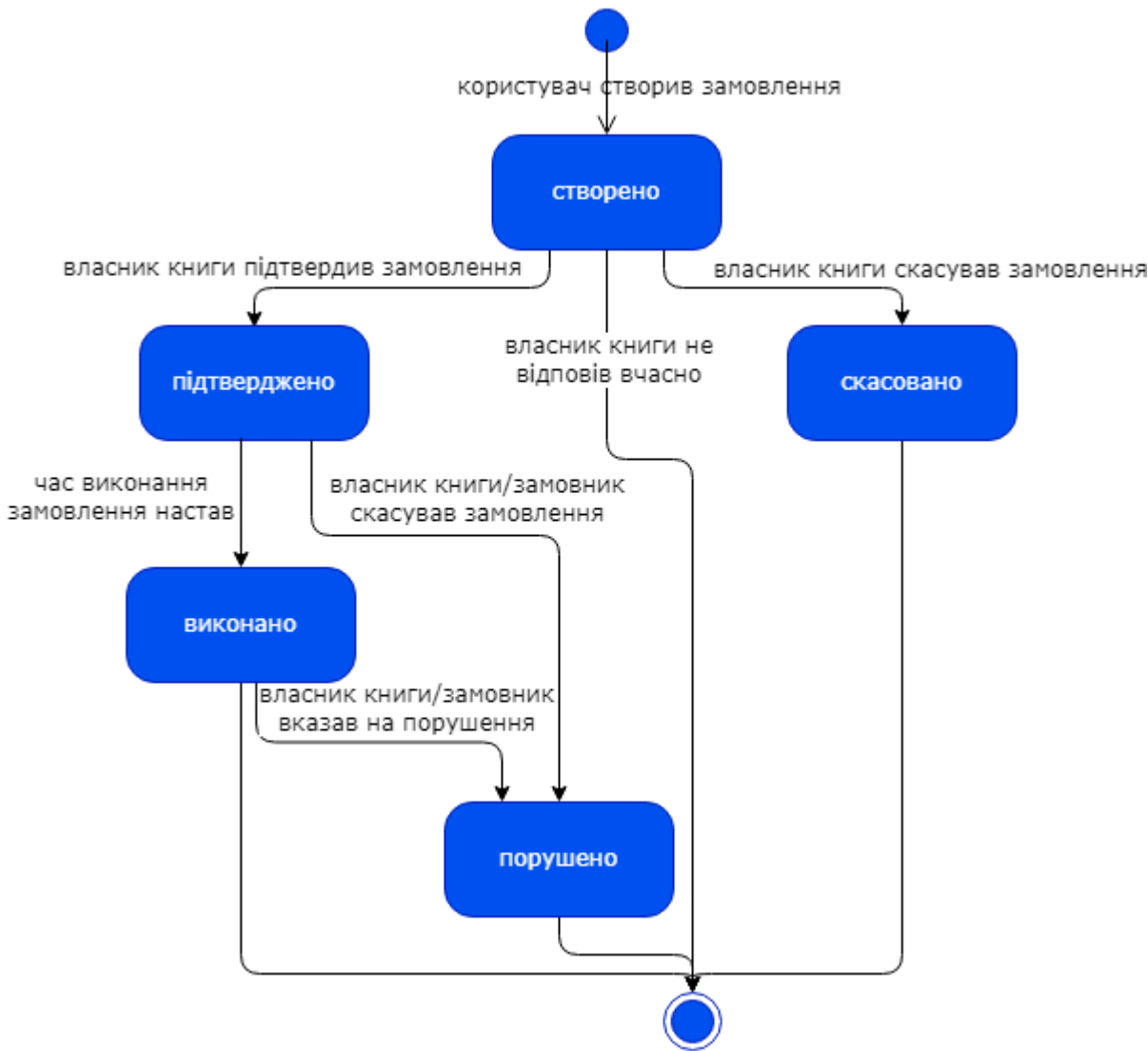


Рисунок 3.1 – Діаграма станів замовлення подальшого поширення книг або замовлення покупки книги

Замовлення тимчасової оренди передбачає повернення книг після завершення замовлення, а тому кількість станів розширюється у такого замовлення (рис. 3.2).

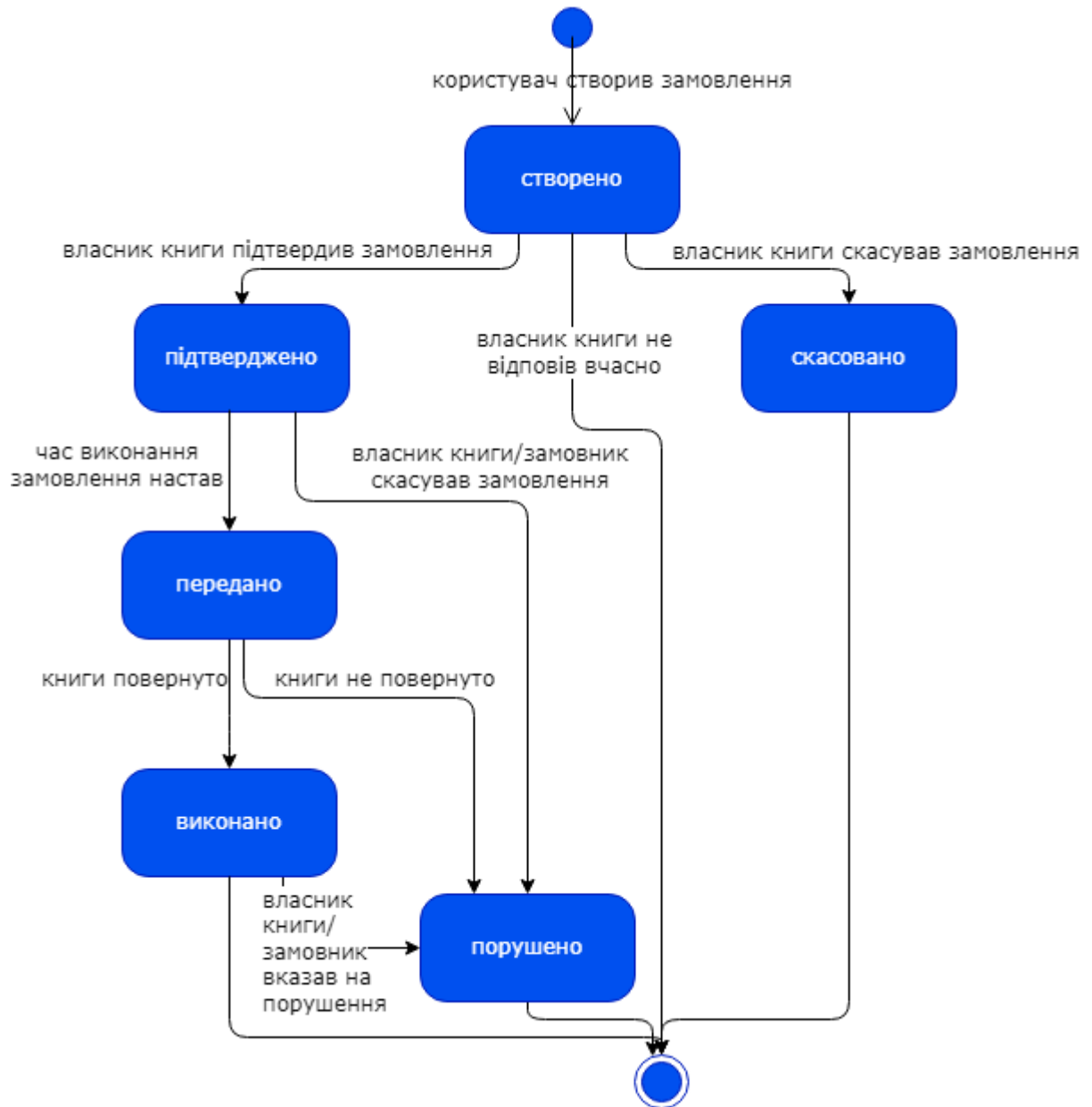


Рисунок 3.2 – Діаграма станів замовлення тимчасової оренди книг

Замовлення обміну книг передбачає деталізацію початкової роботи, адже в ньому одна зі сторін може змінювати склад замовлення, пропонуючи інші книги, тому це замовлення може знаходитися в стані частково підтвердженого (рис. 3.3).

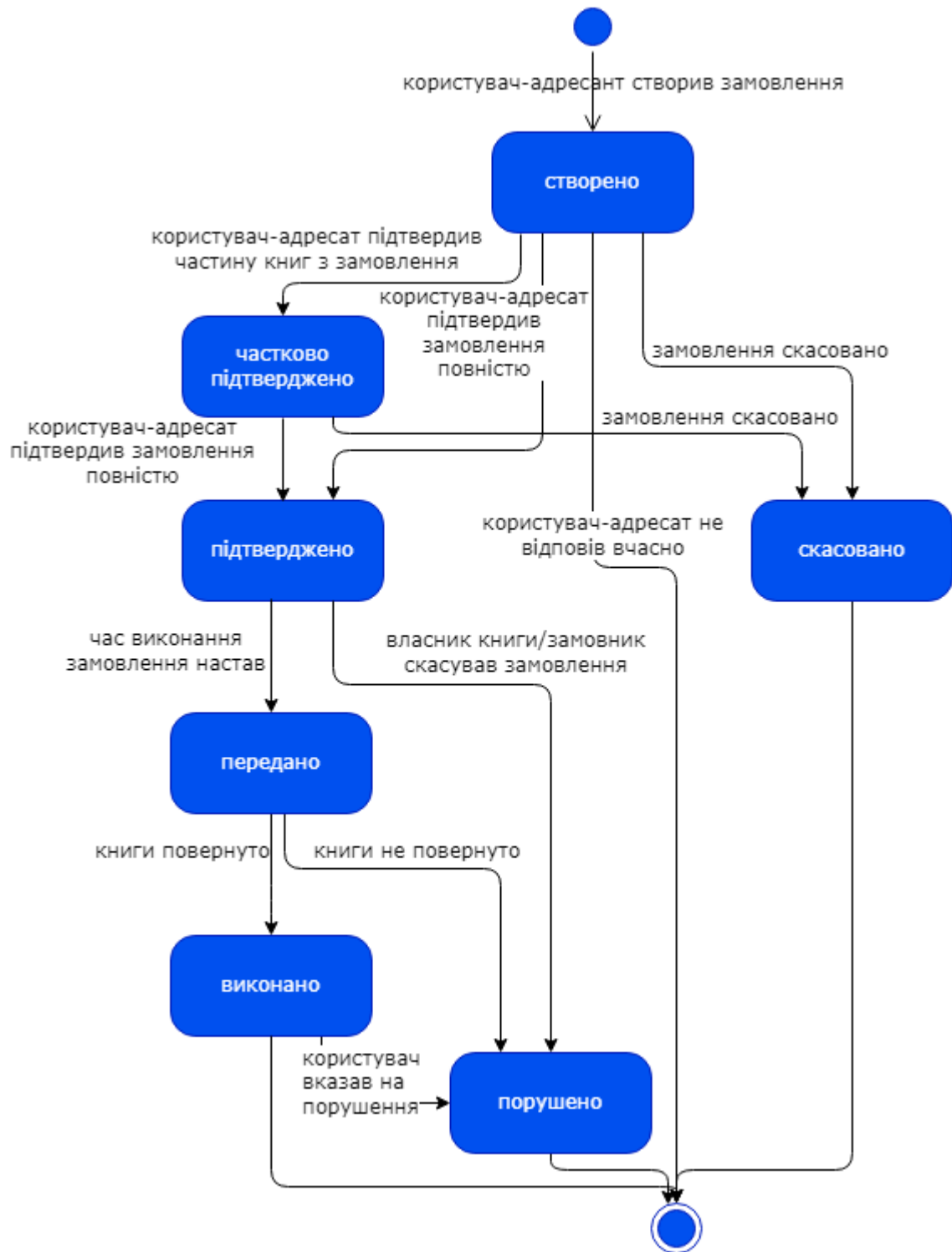


Рисунок 3.3 – Діаграма станів замовлення обміну книг

3.2 Реалізація функцій додатку

Пошук екземплярів книжок для обміну реалізовано через уявлення `find_book_instances_view(request)`. Параметри запиту використовуються як

параметри пошуку, включаючи стан екземпляру книги (`condit`), інтервал усередненої оцінки (від `lreview` до `rreview`), можливість придбання (будь-яке значення в параметрі `buy`), відсутність необхідності внесення застави (будь-яке значення в параметрі `deposit`), варіант обміну (`share`: 0 – тимчасово зі сплатою, 1 – відкрите поширення, 2 – обмін книг, 3 – тимчасово зі сплатою або обмін), припустимість використання доставки (будь-яке значення в параметрі `deliv`), інтервал стану екземпляру (від `lcondit` до `rcondit`), письменник (`writer`), назва книги (`book`), жанри (`genres`), теги (`tags`), видавець (`publisher`), мова книги (`language`), тип книги (`type`), інтервал публікації (від `lyear` до `ryear`), період, на який потрібна книга (від `lsharedate` до `rsharedate`), сума застави в заданому інтервалі (від `ldeposit` до `rdeposit`), вартість обміну (від `lpay` до `rpay`). За кожним параметром виконується відфільтровування екземплярів, які не відповідають встановленим значенням. Окрім того за відібраними екземплярами перевіряється, чи вони знаходяться на відстані не більшій ніж та, яку в своєму профілі визначив користувач, та чи доступні ці екземпляри на необхідний період часу, тобто що відсутні по цим екземплярам замовлення на вказані дати. Усі екземпляри, які проходять цю перевірку, перевіряються далі на вартість та суму застави. Якщо і за вартістю, і за сумою застави ці екземпляри проходять перевірку за вказаним інтервалом, то такі екземпляри книг сортуються за усередненою оцінкою з відгуків і передаються в шаблон вебсторінки `find_book_instances.html`, створений на основі CSS-шаблону [13].

Пошук книжок реалізовано через уявлення `find_books_view(request)`. Параметри пошуку передаються через параметри запиту, включаючи письменника, назву книги, теги, жанри, видавця, мову книги, тип книги, рік або інтервал року виробництва. Останні параметри належать до видання, а перші – до книги загалом. Тому на основі першої спочатку відбираються книги, які підпадають під запит. Якщо параметри з другої частини не задані, то повертається тільки загальний список книг. Якщо зустрічаються значення з другої половини параметрів, то до них у результуючому контексті додається

за ключем `publishedbooks` список видань. Уявлення використовує шаблон вебсторінки `find_books.html`.

Пропонування книжок за зацікавленостями користувача реалізовано через уявлення `suggest_bookinstances_view(request)` з підтримкою шаблону вебсторінки `suggest.html`. Формуються 2 списки, що повертаються у відповідних ключах словника контексту `writer_books` і `published_books` у відповідності з моделями для авторизованого користувача. Під час пошуку перевіряється, чи знаходиться екземпляр книги на відстані не більший ніж у моделі, якщо не допускається використання доставки для цієї зацікавленості.

Внесення власного екземпляру книжки для обміну реалізовано через уявлення `add_book_view(request)`. У всіх уявленнях внесення даних реалізовано подібну логіку, яка передбачає використання в основі форми – у даному випадку форми `BookinstanceForm`. При зверненні на основі методу `post` дані витягаються з полів форми, виконуючи при цьому перевірку даних, вносячи дані про авторизованого користувача в якості власника екземпляра книги та зберігаючи дані в БД. За допомогою пакету `uuid` при цьому генерується унікальний номер, який може використовуватися для ідентифікації книги в подальшому, якщо власник згоден нанести цей номер на книгу. При зверненні на основі методу `get` завантажується порожня форма.

Керування внесеними екземплярами книжок користувача реалізовано через уявлення `edit_book_view(request, id)` для редагування даних екземпляра книги з ідентифікатором `id` на основі тієї ж форми `BookinstanceForm` і тієї ж вебсторінки (за методу `get` у випадку, якщо авторизований користувач є власником екземпляра книги, форма заповнюється даними екземпляра книги) та уявлення `book_instances_view(request)` для відтворення списку екземплярів книг авторизованого користувача.

Внесення екземпляру відсутньої в системі книги реалізовано через уявлення `add_new_book_instance_view(request)`. В основі використовується одразу декілька форм, адже при такому внесенні вноситься не тільки екземпляр книги, але і книга, і видання, якщо вони відсутні в БД. До цих форм

належать BookForm для внесення книги, PublishedForm для внесення видання та BookinstanceForm для внесення самого екземпляру книги. Шаблоном вебсторінки є `add_my_book.html`.

Відтворення списку екземплярів видання книги реалізовано через уявлення `publishedbook_instances_view(request, publishedbook_id = None)`, у якому шаблон вебсторінки `publishedbook_instances.html` наповнюється даними самого видання книги з номером `publishedbook_id` через ключ `book` та безпосередньо екземплярами книг через ключ `instances`. Однак, за другим ключем потрапляють не всі екземпляри, а тільки ті, які розташовані на відстані не більшій, ніж та, яка вказана максимальною в налаштуваннях авторизованого користувача, відносно базової адреси користувача.

Замовлення тимчасової оренди книг реалізовано через уявлення `add_shareorder_view(request)` з підтримкою форми `ShareorderForm` і шаблонів вебсторінок `shareorder.html`, `payment.html`, `result.html`.

Замовлення подальшого поширення книг реалізовано через уявлення `add_openshareorder_view(request)` з підтримкою форми `OpenshareorderForm` і шаблону вебсторінки `openshareorder.html`.

Замовлення обміну книг реалізовано через уявлення `add_changeorder_view(request)` з підтримкою форми `ChangeorderForm` і шаблону вебсторінки `changeorder.html`.

Керування станом замовлення реалізовано через уявлення `set_order_stage_view(request, ordertype = 0, order_id = None, stage)`, де через параметр `order_id` вказується ідентифікатор замовлення, а через параметр `ordertype` конкретний тип замовлення (тимчасове, обмін, подальше поширення, покупка). При керуванні замовленнями перевіряється роль авторизованого користувача у замовленні, встановлюється поточний стан і в залежності від поточного стану і значення стану, вказаного в параметрі `stage`, реалізується перехід до потрібного (наступного) стану у відповідності з описаною в попередньому підрозділі логікою. Якщо це некоректний стан, то перехід не відбувається. Якщо це стан зі скасування, а замовлення є

сплаченим, то повертається внесена сума на базований рахунок користувача. Якщо замовлення передбачало внесення застави, а тепер воно завершено, то знову ж таки повертається застава на вказаний рахунок.

Відтворення списку створених замовлень користувачем реалізовано через уявлення `account_orders_view(request)`, при цьому автором замовлень визначається авторизований користувач, а відібрані замовлення передаються у шаблон вебсторінки `orders.html`.

Відтворення списку замовлень щодо книг користувача реалізовано через уявлення `owner_orders_view(request)`, при цьому відбираються всі замовлення, що стосуються книг, якими на даний момент володіє авторизований користувач, а відібрані замовлення передаються у шаблон вебсторінки `owner_orders.html`.

Відтворення списку замовлень за екземпляром книги реалізовано через уявлення `bookinstance_orders_view(request, book_id = None)`, при цьому замовлення виділяються для екземпляру книги з номером `book_id`, для якого перевіряється, що авторизований користувач є його власником. Якщо це так, то шаблон вебсторінки `bookinst_orders.html` заповнюється замовленнями за цим екземпляром.

Придбання книги реалізовано через уявлення `add_buyorder_view(request)` з підтримкою форми `BuyorderForm` і шаблонів вебсторінок `buyorder.html`, `payment.html`, `result.html`.

Внесення власної інформації користувача реалізовано через уявлення `add_account_view(request)` на основі форми `AccountForm` і шаблону вебсторінки `my_profile.html`.

Керування власною інформацією користувача реалізовано через уявлення `edit_account_view(request)` для оновлення даних через шаблон вебсторінки `my_profile.html` і форму `AccountForm` та через уявлення `account_view(request, acc_id = None)` для відображення даних користувача з номером `acc_id` або для авторизованого користувача, якщо номер не вказано, використовуючи при цьому шаблон вебсторінки `profile.html`.

Відтворення інформації про письменника реалізовано через уявлення `writer_view(request, id = None)`, в якому `id` задає ідентифікатор письменника, за яким витягаються дані про нього самого та про 4 останніх книжок, які він написав і опублікував, формуючи контекстний словник з ключами `writer` і `books` відповідно для шаблону вебсторінки `writer.html`.

Відтворення списку книжок письменника реалізовано через уявлення `writer_books_view(request, writer_id = None)`, в якому `writer_id` задає ідентифікатор самого письменника, а шаблон вебсторінки `writer_books.html` заповнюється даними книжок письменника, відсортованими за датою публікації, починаючи з останньої.

Відтворення рейтингу видань книжок за жанром реалізовано через уявлення `genre_books_view(request, genre_id = None)`, в якому `genre_id` задає ідентифікатор жанру, а шаблон вебсторінки `genre_books.html` заповнюється даними видань книжок, які належать вказаному жанру, сортуючи їх всі за усередненою оцінкою рецензій за зменшенням.

Відтворення рейтингу видань книжок за тегом реалізовано через уявлення `tag_books_view(request, tag_id = None)`, в якому `tag_id` задає ідентифікатор тегу, а шаблон вебсторінки `tag_books.html` заповнюється даними видань книжок, які належать вказаному тегу, сортуючи їх всі за усередненою оцінкою рецензій за зменшенням.

Відтворення видань книги реалізовано через уявлення `book_publishes_view(request, book_id = None)`, в якому `book_id` задає ідентифікатор книги, а шаблон вебсторінки `published_books.html` заповнюється даними письменників у ключі `writers` словника і списком опублікованих видань, відсортованих за датою публікації, починаючи з останніх, у ключі `books` словника.

Відтворення всіх рецензій за книгою реалізовано через уявлення `publishedbook_critiques_view(request, pub_id = None)`, де шаблон вебсторінки `pub_critiques.html` заповнюється списком всіх рецензій за виданням книги з

номером `pub_id`, відсортованим за часом створення, та даними самого видання книги.

Відтворення всіх прорецензованих видань користувачем реалізовано через уявлення `account_critiques_view(request, account_id = None)`, де шаблон вебсторінки `account_critiques.html` отримує контекстний словник з ключами `name` і `last_name`, де вказано ім'я та прізвище користувача з номером `account_id`, а його список рецензій формує словник, у якому ключем є видання, а значення за ключем – список рецензій авторизованого користувача про це видання. За відсутності номеру в якості параметра відбувається відбір для авторизованого користувача.

Відтворення всіх відгуків користувача реалізовано через уявлення `account_reviews_view(request, account_id = None)`, де шаблон вебсторінки `account_reviews.html` отримує контекстний словник з даними відгуків, відсортований за часом створення. За відсутності номеру в якості параметра відбувається відбір для авторизованого користувача.

Відтворення всіх відгуків про користувача реалізовано через уявлення `shared_reviews_view(request, account_id = None)`. В якому шаблон вебсторінки `account_reviews.html` заповнюється відгуками про користувача з номером `account_id` або про авторизованого користувача, якщо номер не задано.

Відтворення всіх відгуків про екземпляр книги реалізовано через уявлення `bookinstance_reviews_view(request, book_id = None)`, де шаблон вебсторінки `bookinstance_reviews.html` наповнюється даними відгуків, які були створені стосовно екземпляру книги з номером `book_id`.

Відтворення інформації про книгу реалізовано через уявлення `book_view(request, book_id = None)`, де шаблон вебсторінки `book.html` наповнюється даними про саму книгу (ключ `book`) з номером `book_id`, про 4 її останні за часом публікації видання (ключ `published_books`) та 4 останні за часом створення рецензії до одного з видань книги (ключ `critiques`).

Відтворення інформації видання книги реалізовано через уявлення `publishedbook_view(request, pub_id = None)`, де шаблон вебсторінки

publishedbook.html наповнюється даними про саме видання (ключ book) з номером pub_id, про 4 екземпляри цієї книги, які знаходяться найближче до базової адреси користувача (ключ instances) та 4 останні за часом створення рецензії (ключ critiques).

Відтворення інформації про екземпляр книги реалізовано через уявлення book_instance_view(request, book_id = None), де шаблон вебсторінки book_inst.html наповнюється даними про сам екземпляр книги (ключ book_instance) з номером book_id, про книгу (ключ book), про видання (ключ published_book), про 4 останні відгуки (ключ reviews) та актуальні правила обміну (ключ sharingrules).

Створення заявки на власництво екземпляру книги реалізовано через уявлення add_book_application_view(request, id). Параметр id – ідентифікатор екземпляру книги, на який авторизований користувач висунув заявку на власництво. В основі лежить форма BookinstanceApplicationForm і шаблон вебсторінки book_application.html.

Надсилання повідомлення у співбесіді реалізовано через уявлення account_query_messages_view(request, accquery_id = None), яке окрім форми надсилання повідомлення TextqueryForm також реалізує наповнення контекстного словника списком повідомлень у співбесіді з номером accquery_id, перевіряючи при цьому, що однією зі сторін цієї співбесіди є авторизований користувач. В основі лежить шаблон вебсторінки accounts_messages.html.

Відтворення списку співбесід користувача реалізовано через уявлення account_queries_view(request), у якому для авторизованого користувача запитується список співбесід і даними цього списку наповнюється шаблон вебсторінки queries.html.

Внесення правила оренди книг реалізовано через уявлення add_sharingrule_view(request), в якому на основі шаблону вебсторінки sharing_rule.html відбувається підтримка роботи з формою SharingruleForm.

Керування внесеними правилами оренди книг реалізовано через уявлення `edit_sharingrule_view(request, id)`, в якому на основі шаблону вебсторінки `sharing_rule.html` відбувається підтримка роботи з формою `SharingruleForm`, і `sharingrules_view(request)`, в якому на основі шаблону вебсторінки `my_sharingrules.html` відбувається його наповнення даними правил обміну з накопиченням його за ключем `sharingrules` у словнику даних.

Рецензування видання книги реалізовано через уявлення `add_critique_view(request, pub_id = None)`, в якому на основі шаблону вебсторінки `critique.html` відбувається підтримка роботи з формою `CritiqueForm`.

Внесення відгуку про екземпляр книги реалізовано через уявлення `add_book_review_view(request, book_id = None)`, в якому на основі шаблону вебсторінки `review.html` відбувається підтримка роботи з формою `BookinstanceReviewForm`, але при цьому спочатку перевіряється, що користувач, який авторизований на даний момент, раніше оформлював замовлення стосовно екземпляру книги з номером `book_id` і що це замовлення знаходиться в завершеному стані. Якщо жодного такого замовлення у авторизованого користувача за цією книгою немає, то йому не буде надано доступ до форми відгуку.

Внесення відгуку про користувача реалізовано через уявлення `add_account_review_view(request, acc_id = None)`, в якому на основі шаблону вебсторінки `account_review.html` відбувається підтримка роботи з формою `AccountReviewForm`, але при цьому спочатку перевіряється, що користувач, який авторизований на даний момент, раніше оформлював замовлення у користувача з номером `acc_id` і що це замовлення знаходиться в завершеному стані. Якщо жодного такого замовлення між цими користувачами немає, то йому не буде надано доступ до форми відгуку.

Відтворення списку письменників за країною реалізовано через уявлення `country_writers_view(request, country_id = None)`, де відбувається заповнення шаблону вебсторінки `country_writers.html` даними письменників,

які проживають в країні з номером `country_id` або які народилися в місті, що знаходиться в країні з цим номером.

Відтворення списку письменників за містом реалізовано через уявлення `city_writers_view(request, city_id = None)`, де відбувається заповнення шаблону вебсторінки `city_writers.html` даними письменників, які народилися в місті з номером `city_id`, а за ключем `country_cities` у словник вноситься набір міст за кожною країною.

Відтворення книг, виданих видавництвом реалізовано через уявлення `publisher_books_view(request, publisher_id = None)`, в якому шаблон вебсторінки `publisher_books.html` наповнюється даними виданих книжок за видавництвом з номером `publisher_id`.

Відзначення зацікавленості користувача у виданні реалізовано через уявлення `add_publishedbook_wanting_view(request, pub_id = None)`, в якому на основі шаблону вебсторінки `pubbook_wanting.html` відбувається підтримка роботи з формою `PublishedbookWantingForm`.

Відзначення зацікавленості користувача у книгах письменника реалізовано через уявлення `add_writer_wanting(request, writer_id = None)`, в якому на основі шаблону вебсторінки `writer_wanting.html` відбувається підтримка роботи з формою `WriterWantingForm`.

3.3 Висновки за розділом 3

Усі реалізовані функції за визначеними для програми сценаріями представлені у вигляді програмних функцій у складі комбінації уявлення і шаблонів за структурою вебфреймворку Django.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис програми

Програма призначена для обміну книгами як шляхом надання їх власником у тимчасове користування іншим людям за сплату або без сплати з внесенням або без внесення застави за це, так і шляхом передавання їх орендарю без необхідності повернення, реалізацією обміну книг одного власника на книги іншого власника з підтримкою подальшого придбання книги за згоди обох сторін.

4.2 Експлуатація програми

Програма побудована для надання під час взаємодії користувачу будь-яких функціональних можливостей, доступних загалом, але якщо користувач не має права їх використовувати, то йому відображається сторінка авторизації. Наприклад, якщо користувач не виконав авторизацію, але спробував відобразити через меню сторінку з замовленнями, то його буде перенаправлено на сторінку авторизації.

Загальне меню структурує доступ до основних функцій. Пункт «Книги» дозволяє виконувати пошук видань книг, задаючи у формі необхідні критерії пошуку, після чого відображаються результати зі списком видань. З кожного результату можна відкрити відповідну сторінку з повною інформацією, можна перейти до сторінки книги загалом або до сторінки письменника (рис. 4.1), який є автором книги. На сторінці письменника виводиться його фотографія, прізвище та ім'я, загальна інформація, інформація про дату та місце народження. Посилання «Цікавить» надає можливість внести письменника до зацікавленостей. Для таких письменників з меню «Персональна» відображаються екземпляри книжок, які знаходяться поруч з авторизованим користувачем. Також там виводяться видання книжок, за якими таким же чином визначив свою зацікавленість користувач.

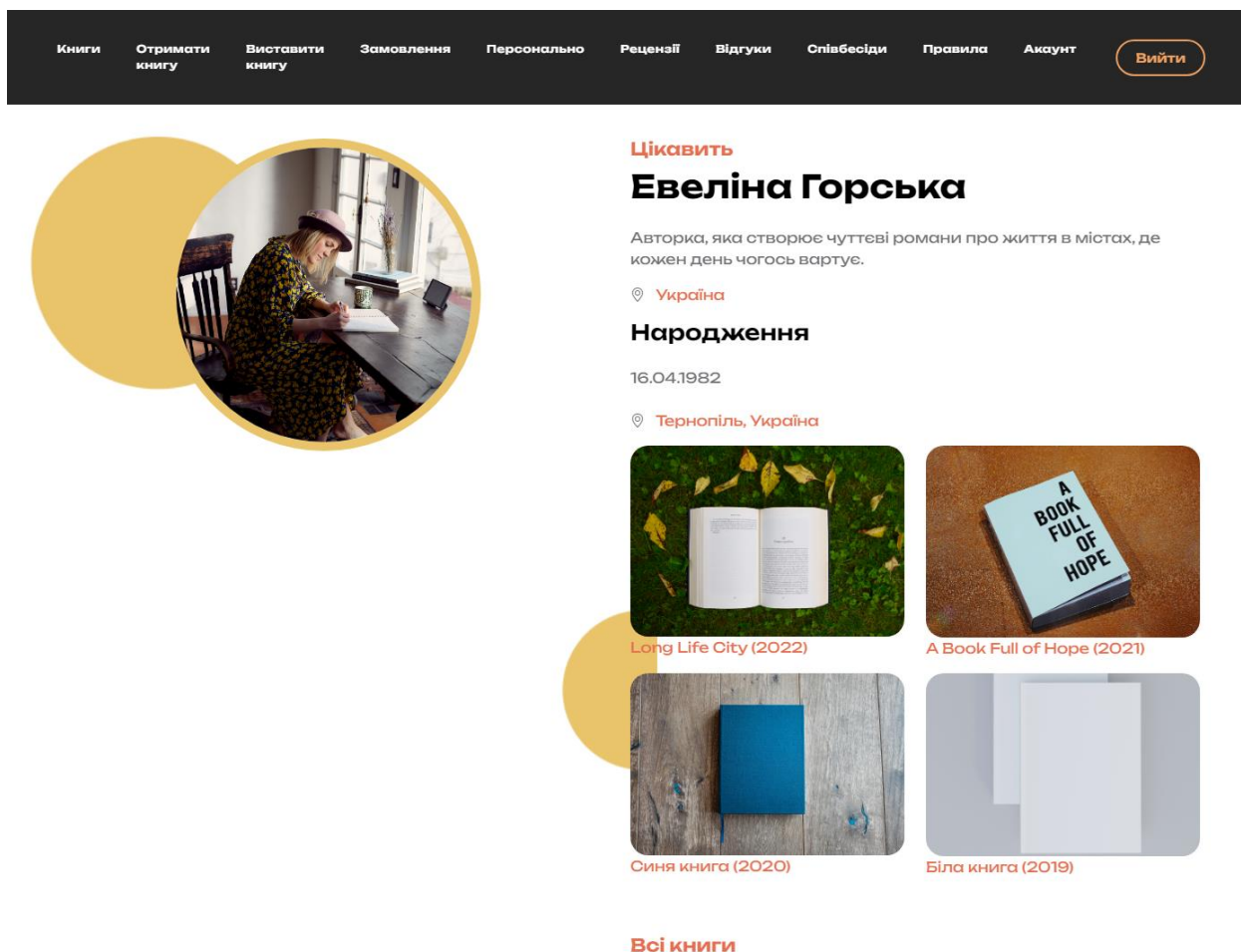


Рисунок 4.1 – Вебсторінка письменника

На сторінці письменника відображаються його останні 4 видані книжки. Натискаючи на назву книги, можна перейти до її сторінки, а там вже побачити видання цієї книги. За виданням відповідно можна побачити екземпляри. Або через меню «Отримати книгу» відкривається пошук екземплярів книжок, звідки також можна за встановленими критеріями знайти потрібну книгу, а тоді перейти до її сторінки.

На сторінці екземпляру книги (рис. 4.2) відображається автор або автори книги з можливістю перейти на їхні сторінки, назва книги, дані про власника (також є посиланням, можна відкрити сторінку його як користувача).

Якщо користувач бажає замовити книгу, то він має натиснути на кнопку «Замовити». Вона дозволяє зберегти книгу в тимчасовий кошик. При цьому ще не визначається тип замовлення. Дані накопичуються, а потім, якщо відкрити меню «Замовлення», то на ній буде відображено перелік всіх книг,

які було замовлено, там можна буде обрати тип замовлення, в залежності від якого відображається певна форма для вибору параметрів замовлення. Після того, як потрібні дані внесені, якщо таке замовлення можна виконати, то його можна створити. На цій же сторінці відображається посилання з переходом до всіх власних замовлень.

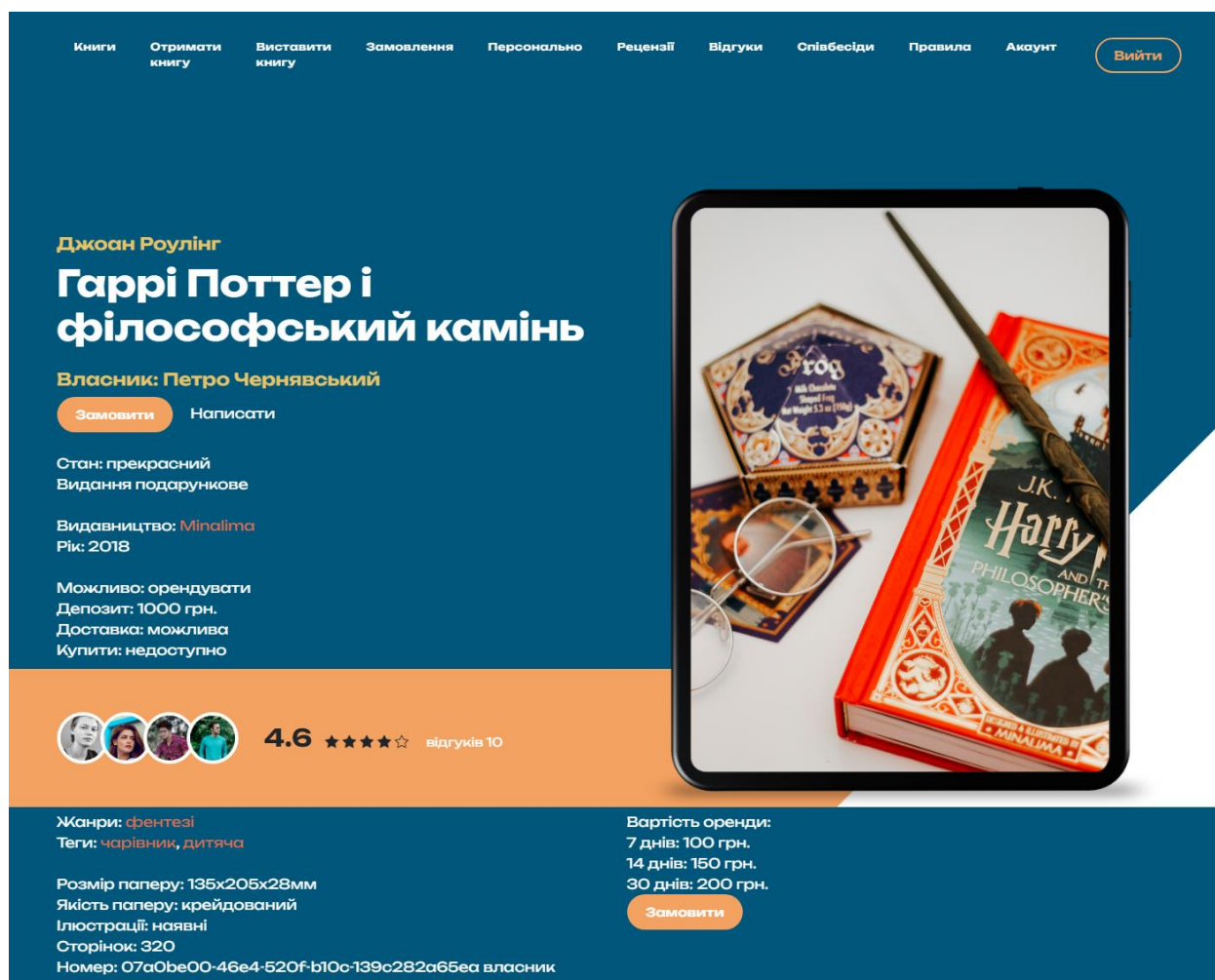


Рисунок 4.2 – Вебсторінка екземпляру книги

На сторінці екземпляру книги відображається інформація про її стан, тип видання, видавництво, рік видання, тип замовлень, які можна виконати, депозит, який в разі оренди потрібно внести, можливість доставки, можливість придбання книги. Також виводиться середня оцінка екземпляру книги, фотографії 4 останніх авторів відгуків. Натискаючи на кількість відгуків, відображаються всі відгуки за екземпляром книги. Нижче на сторінці також

відображається текст 4 останніх відгуків. Перед цим відображаються жанри, теги з можливістю перейти на сторінку з найкращими виданнями за цим жанром або тегом, інформація про розмір та якість паперу, наявність ілюстрацій, кількість сторінок, унікальний згенерований номер книги. Посилання «власник» біля номеру дозволяє сформулювати заявку на те, що користувач тепер є власником цього екземпляру. Також відображається інформація про вартість оренди цієї книги на різні періоди часу, а також про вартість придбання цієї книги, якщо воно доступне.

Вебсторінка власних замовлень (рис. 4.3) містить додаткове меню з вибором типу замовлення. Під кожне замовлення виводиться його стан, періоди роботи над ним, інформація про екземпляри книжок у замовленні. Якщо замовлення закінчено, то можна створити відгук за екземпляром книги та за користувачем, який є власником книги.

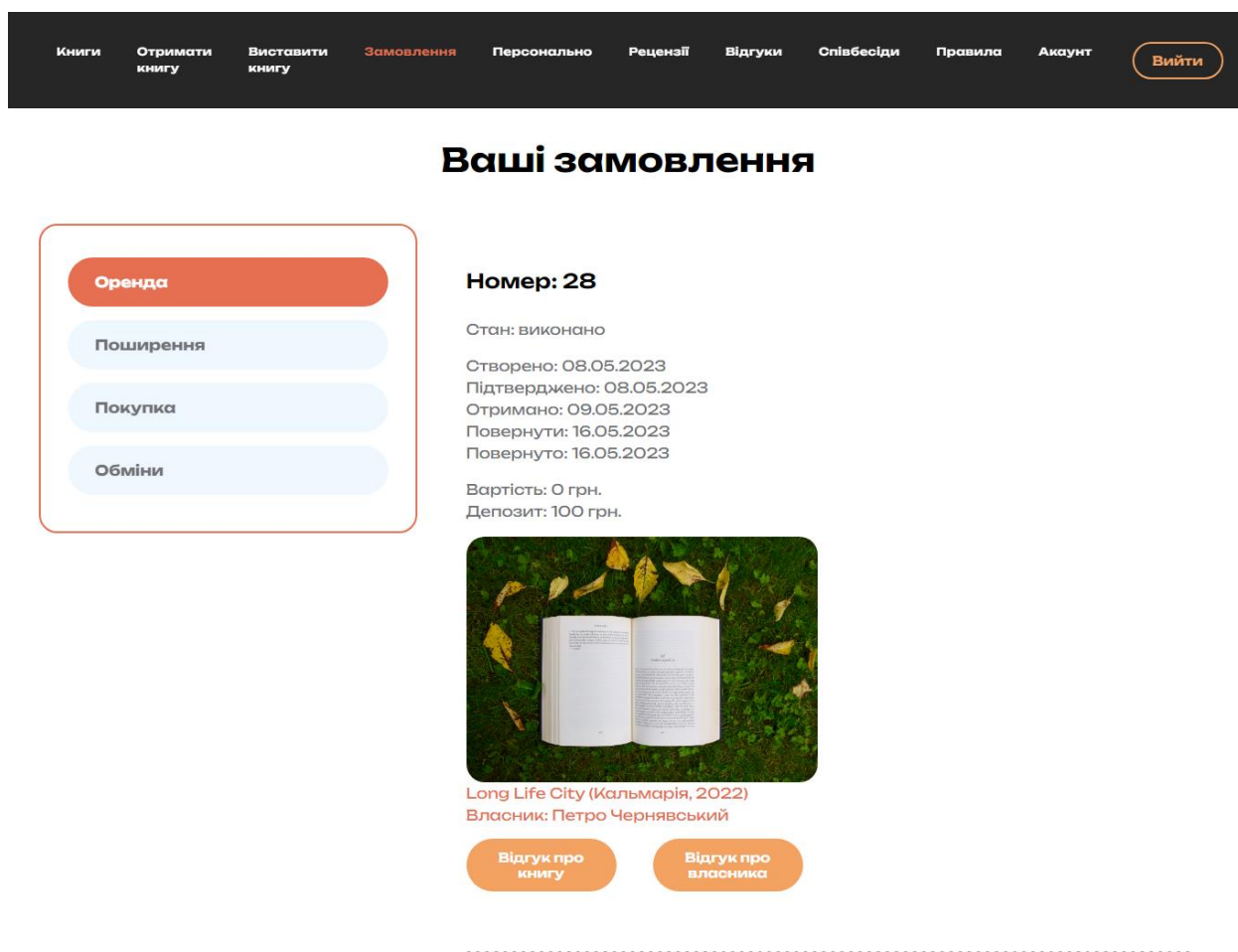


Рисунок 4.3 – Вебсторінка власних замовлень

Для замовлення у інших станах відображається замість кнопки для внесення відгуку кнопка для відповідної дії у відповідності з діаграмою станів.

Зі сторінки користувача можна переглянути відгуки про нього (рис. 4.4).

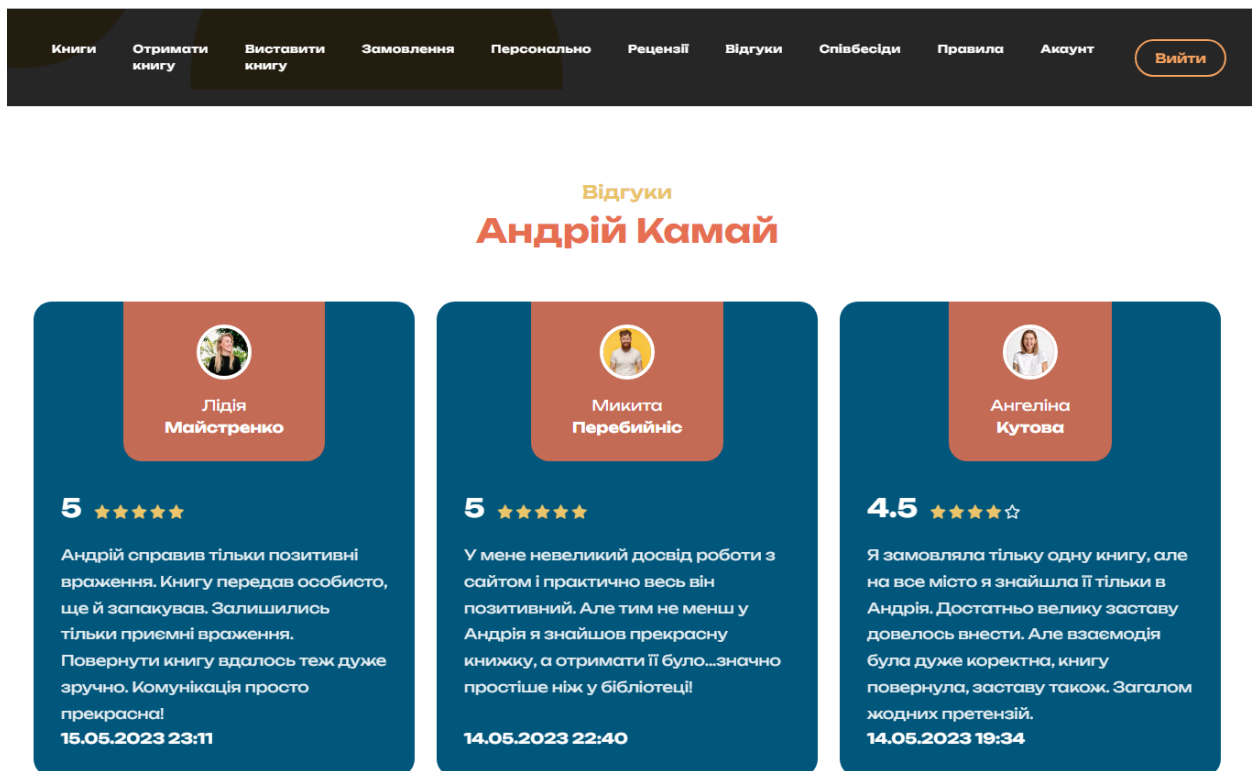


Рисунок 4.4 – Вебсторінка списку відгуків користувача

Пункт меню «Виставити книгу» дозволяє переглянути список всіх власних екземплярів книг, які виставив для замовлення авторизований користувач. Для внесення нового екземпляру потрібно ще раз натиснути на кнопку «Виставити книгу» на сторінці. Біля кожної власної книги надається можливість редагувати дані про неї, переходити до замовлень за цією книгою. За окремим посиланням «Замовлення» на цій сторінці можна перейти до замовлень за всіма книгами авторизованого користувача. З цих сторінок замовленнями можна керувати через кнопки, які надаються у відповідності з діаграмами станів. Визначити правила (вартість) оренди можна через меню «Правила».

Створювати рецензії про видання книг можна через меню «Рецензії».

Кожному користувачу, який є власником книги, можна написати (на сторінці відображається посилання «Написати»). Переглянути перелік співбесід можна через меню «Співбесіди».

4.3 Тестування програми

Усі функції, які було заплановано для реалізації, було перевірено на відповідність отриманих результатів очікуваним (табл. 4.1). Перевірка пройдено успішно.

Таблиця 4.1 – Результати тестування програми

Сценарій	Підсумок
1	2
Пошук екземплярів книжок для обміну	Успішно
Пошук книжок	Успішно
Пропонування книжок за зацікавленостями користувача	Успішно
Внесення власного екземпляру книжки для обміну	Успішно
Керування внесеними екземплярами книжок користувача	Успішно
Внесення екземпляру відсутньої в системі книги	Успішно
Відтворення списку замовлень за екземпляром книги	Успішно
Відтворення списку екземплярів видання книги	Успішно
Замовлення тимчасової оренди книг	Успішно
Замовлення подальшого поширення книг	Успішно
Замовлення обміну книг	Успішно
Керування станом замовлення	Успішно
Відтворення списку створених замовлень користувачем	Успішно
Відтворення списку замовлень щодо книг користувача	Успішно
Придбання книги	Успішно

Продовження таблиці 4.1

1	2
Внесення власної інформації користувача	Успішно
Реєстрація користувача	Успішно
Керування власною інформацією користувача	Успішно
Керування адресами користувача	Успішно
Відтворення інформації про письменника	Успішно
Відтворення інформації про користувача	Успішно
Відтворення списку книжок письменника	Успішно
Відтворення рейтингу видань книжок за жанром	Успішно
Відтворення рейтингу видань книжок за тегом	Успішно
Відтворення видань книги	Успішно
Відтворення всіх рецензій за книгою	Успішно
Відтворення всіх прорецензованих видань користувачем	Успішно
Відтворення всіх відгуків користувача	Успішно
Відтворення всіх відгуків про користувача	Успішно
Відтворення всіх відгуків про екземпляр книги	Успішно
Відтворення інформації про книгу	Успішно
Відтворення інформації видання книги	Успішно
Відтворення інформації про екземпляр книги	Успішно
Створення заявки на власництво екземпляру книги	Успішно
Створення співбесіди з користувачем	Успішно
Надсилання повідомлення у співбесіді	Успішно
Відтворення списку співбесід користувача	Успішно
Внесення правила оренди книг	Успішно
Керування внесеними правилами оренди книг	Успішно
Рецензування видання книги	Успішно
Внесення відгуку про екземпляр книги	Успішно
Внесення відгуку про користувача	Успішно

Кінець таблиці 4.1

1	2
Відтворення списку письменників за країною	Успішно
Відтворення списку письменників за містом	Успішно
Відтворення книг, виданих видавництвом	Успішно
Відзначення зацікавленості користувача у виданні	Успішно
Відзначення зацікавленості користувача у книгах письменника	Успішно

4.4 Висновки за розділом 4

Реалізовану програму було описано з представленням основної структури роботи користувача з програмою, деталізованим виділенням основних функціональних можливостей, а функціональні можливості було відтестовано з доведенням до успішних результатів за кожним з запланованих сценаріїв.

ВИСНОВКИ

Проведений огляд і аналіз ПЗ, що може використовуватися для обміну книгами, дозволив визначити концепцію ПЗ, що розроблялось в роботі, включаючи можливості рецензування видань і екземплярів книг, обміну книгами на основі як надання у тимчасове використання з можливістю внесення заставу, так і надання у подальше безкоштовне поширення, на основі обміну книгами і придбання за потреби. Результати аналізу відповідають і підтверджують мету роботи, яка полягала в розробці ПЗ обміну книгами для підтримки можливостей продовження спільного використання книг після їхнього першого придбання, що було досягнуто в роботі.

Визначено структуру БД програми з виділенням сутностей книги, видання книги, екземпляра книги, різних варіантів замовлень, та відповідні функції і сценарії роботи користувача з цими функціями. Програму реалізовано мовою Python за допомогою засобів веброзробки, включаючи HTML, CSS, вебфреймворк Django, та системи MySQL для роботи з базою даних. Усі реалізовані функції за визначеними для програми сценаріями представлені у вигляді програмних функцій у складі комбінації уявлення і шаблонів за структурою вебфреймворку Django.

Створена програма надає можливість не тільки передати тимчасово книгу в користування іншій людині, але і фактично підтримувати розширення її подальшого використання за рахунок страхування інтересів власника на основі внесення застави, за рахунок надання гнучкої системи варіантів обміну та надання інформації про орендарів і конкретні екземпляри книг, а також конкретні видання книг на основі використання рецензування і відгуків.

Реалізовану програму було описано з представленням основної структури роботи користувача з програмою, деталізованим виділенням основних функціональних можливостей, а функціональні можливості було відтестовано з доведенням до успішних результатів за кожним з запланованих сценаріїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Curcic, D. Book Sales Statistics – WordsRated [Electronic resource] / D. Curcic. – Access mode : <https://wordsrated.com/book-sales-statistics/#:~:text=The%20global%20market%20for%20second,book%20market%2C%20as%20of%202020.>
2. Talbot, D. Used Book Sales Statistics [Electronic resource] / D. Talbot. – Access mode : <https://wordsrated.com/used-books-statistics/>.
3. Second Hand Books Market ending up at a valuation of US\$ 45.53 Bn by the end of 2032, Persistence Market Research [Electronic resource]. – Access mode : <https://www.globenewswire.com/news-release/2022/04/06/2417884/0/en/Second-Hand-Books-Market-ending-up-at-a-valuation-of-US-45-53-Bn-by-the-end-of-2032-Persistence-Market-Research.html>.
4. Bhula, P. Essay [Electronic resource] : The enduring popularity of second hand books / P. Bhula. – Access mode : <https://www.hindustantimes.com/books/essay-the-enduring-popularity-of-second-hand-books-101646821839244.html>.
5. Goodreads. Meet your next favorite book [Electronic resource]. – Access mode : <https://www.goodreads.com/>.
6. Take a Book. Share a Book. Little Free Library [Electronic resource]. – Access mode : <https://littlefreelibrary.org/>.
7. Share a Book [Electronic resource]. – Access mode : <https://shareabook.lk/>.
8. Share A Book. Facebook [Electronic resource]. – Access mode : <https://www.facebook.com/shareabook.lk/>.
9. Our Documentation. Python.org [Electronic resource]. – Access mode : <https://www.python.org/doc/>.
10. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.
11. Welcome to Pyramid, a Python Web Framework [Electronic resource].

– Access mode : <https://trypython.com/>.

12. Django vs. Pyramid – comparing Python Web Frameworks [Electronic resource]. – Access mode : <https://sunscrapers.com/blog/django-vs-pyramid-comparing-a-python-web-frameworks/>.

13. Ebook Landing Template [Electronic resource]. – Access mode : <https://templatemo.com/tm-588-ebook-landing>.

ДОДАТОК А
Технічне завдання

Вступ

Книги, як і інші товари, продовжують своє життя після першого придбання. Для книг відомою є процедура буккросингу, коли власники вирішують дати нове життя книзі і розповсюджувати її між людьми, яким цікаво її почитати. Як і будь-який інший товар, придбану раніше книгу можна повторно продати. Це ще один шлях подальшого життя книги. Але це все не єдині напрямки. Людина може мати книгу, яку більше не використовує активно, проте хоче її мати в бібліотеці, але на певний період може поділитися нею з іншими людьми. Однак, власника може зупиняти те, що книга може бути вкрадена або пошкоджена. Відгуки про орендарів, внесення застави може цю проблему вирішити разом з наданням певної мережі. Для підтримки цієї мережі користувачів і має використовуватися програма, яку потрібно розробити в роботі.

A.1 Підстави для розробки

Завдання на дипломну кваліфікаційну роботу сформульовано у відповідності з наказом № 87 від 4 квітня 2023 р. за Національним університетом «Запорізька політехніка», за яким було визначено тему «Програмне забезпечення обміну книгами».

A.2 Призначення розробки

Програма призначена для обміну книгами як шляхом надання їх власником у тимчасове користування іншим людям за сплату або без сплати з внесенням або без внесення застави за це, так і шляхом передавання їх орендарю без необхідності повернення, реалізацією обміну книг одного власника на книги іншого власника з підтримкою подальшого придбання книги за згоди обох сторін.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Забезпечення призначення програми виконується через реалізацію сценаріїв використання програми на основі наступних функцій:

- пошук книжок для обміну;
- пропонування книжок за зацікавленостями користувача;
- внесення власних екземплярів книжок для обміну;
- керування внесеними екземплярами книжок користувача;
- внесення екземпляру відсутньої в системі книги;
- відтворення списку замовлень за екземпляром книги;
- відтворення списку екземплярів видання книги;
- замовлення книг, включаючи тимчасову оренду книг, подальше поширення, обмін книг;
- керування станом замовлення;
- відтворення списку створених замовлень користувачем;
- відтворення списку замовлень щодо книг користувача;
- відтворення списку замовлень за екземпляром книги;
- придбання книги;
- внесення власної інформації користувача;
- реєстрація користувача;
- керування власною інформацією користувача, включаючи керування адресами користувача;
- відтворення інформації про письменника;
- відтворення інформації про користувача;
- відтворення списку книжок письменника;
- відтворення рейтингу видань книжок за жанром;
- відтворення рейтингу видань книжок за тегом;
- відтворення видань книги;
- відтворення всіх рецензій за книгою;

- відтворення всіх прорецензованих видань користувачем;
- відтворення всіх відгуків користувача;
- відтворення всіх відгуків про користувача;
- відтворення всіх відгуків про екземпляр книги;
- відтворення інформації про книгу;
- відтворення інформації видання книги;
- відтворення інформації про екземпляр книги;
- створення заявки на власництво екземпляру книги;
- створення співбесіди з користувачем;
- надсилання повідомлення у співбесіді;
- відтворення списку співбесід користувача;
- внесення правила оренди книг;
- керування внесеними правилами оренди книг;
- рецензування видання книги;
- внесення відгуку про екземпляр книги;
- внесення відгуку про користувача;
- відтворення списку письменників за країною або за містом;
- відтворення книг, виданих видавництвом;
- відзначення зацікавленості користувача у виданні;
- відзначення зацікавленості користувача у книгах письменника.

A.3.2 Умови експлуатації

Експлуатація програми на сервері потребує підтримки, що включає навички роботи з БД, зокрема MySQL, виконання міграцій для Django-додатків та внесення даних через адміністративний інтерфейс при необхідності розширення або зміни письменників, жанрів, видавців або інших каталогів стандартних даних.

А.3.3 Вимоги до складу та параметрів технічних засобів

Набір мінімальних технічних засобів, що мають забезпечуватися на сервері для підтримки роботи вебдодатку, включає:

- процесор не менше ніж Хеон сімейства 3000 або Core 2 Duo;
- оперативна пам'ять обсягом у 4 Гб;
- простір на жорсткому диску з доступним вільним обсягом не менше ніж 3 Гб, але для можливості збереження файлів зображень бажано не менше 20 Гб;
- мережна карта і підключення до мереж Інтернет.

А.4 Порядок контролю та приймання

Для виконання дипломної кваліфікаційної роботи критичним є виконання визначеного завдання, що включає безпосередньо завдання на роботу і технічне завдання, за темою, яка зафіксована наказом, у строки, які визначені завданням на роботу.

У підсумку виконується захист роботи за оприлюдненим наперед графіком перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `book_views.py`

```

from .models import Genre, Book, Writer, Tag,
Publishedbook, Critique, Bookinstance, BookinstanceAddress,
Accountaddress, Language, Shareorder, Changeorder,
Openshareorder, Buyorder, Country, City
from .forms import BookinstanceForm, PublishedForm,
BookForm, BookinstanceApplicationForm, SharingruleForm
from math import radians, cos, sin, atan2, sqrt
from django.db.models import Q

def genre_books_view(request, genre_id = None):
    genre_books = []
    if genre_id:
        genre_get = Genre.objects.get(pk = genre_id, None)
        if genre_get:
            genre_books =
Publishedbook.objects.filter(book__genres =
genre_get).select_related("book").prefetch_related("book_writers")
).order_by('-average_critique')
        return render(request, "booshare/genre_books.html",
{"books": genre_books})

def writer_books_view(request, writer_id = None):
    writer_books = []
    if writer_id:
        writer_get = Writer.objects.get(pk = writer_id,
None)

        if writer_get:
            writer_books = Book.objects.filter(writers =
writer_get).prefetch_related("genres").order_by('-
first_published')
        return render(request, "booshare/writer_books.html",
{"books": writer_books})

def tag_books_view(request, tag_id = None):
    tag_books = []
    if tag_id:
        tag_get = Tag.objects.get(pk = tag_id, None)
        if tag_get:
            tag_books =
Publishedbook.objects.filter(book__tags =
tag_get).select_related("book").prefetch_related("book_writers").
order_by('-average_critique')
        return render(request, "booshare/tag_books.html",
{"books": tag_books})

```

```

def book_publishes_view(request, book_id = None):
    publishedbooks = []
    if book_id:
        book_get = Book.objects.get(pk = book_id, None)
        writers = Writer.objects.filter(books = book_get)
        if book_get:
            publishedbooks =
Publishedbook.objects.filter(book =
book_get).prefetch_related("languages").select_related("book").o
rder_by('-first_published_year')
        return render(request, "booshare/published_books.html",
{"books": publishedbooks, "writers": writers})

def book_view(request, book_id = None):
    book_get = None
    published_books = []
    critiques = []
    if book_id:
        book_get =
Book.objects.prefetch_related("writers").prefetch_related("genre
s").prefetch_related("tags").get(pk = book_id, None)
        if book_get:
            published_books =
Publishedbook.objects.filter(book = book_get).order_by("-
first_published_year")[:4]
            critiques =
Critique.objects.filter(publishedbook__book =
book_get).order_by("-when_created")[:4]
        return render(request, "booshare/book.html", {"book":
book_get, "published_books": published_books, "critiques":
critiques})

def publishedbook_instances_view(request, publishedbook_id
= None):
    publishedbook_id = None
    published_instances = []
    account = Account.objects.get(id = request.user.id)
    address_from = Accountaddress.objects.filter(account =
account, base = True)[0]
    if publishedbook_id:
        publishedbook_get = Publishedbook.objects.get(pk =
publishedbook_id, None)
        if publishedbook_get:
            published_instances =
Bookinstance.objects.filter(publishedbook = publishedbook_get)
            published_instances_result = []
            for pos, pubbook in published_instances:

```

```

        bookaddress =
BookinstanceAddress.objects.filter(bookinstance =
pubbook).order_by("-when_added")[0]
        delta_long = address_from.address_long -
bookaddress.accountaddress.address_long
        delta_latd = address_from.address_latd -
bookaddress.accountaddress.address_latd
        calc = sin(delta_latd / 2)**2 +
cos(bookaddress.accountaddress.address_latd) *
cos(address_from.address_latd) * sin(delta_long / 2)**2
        val = 2 * atan2(sqrt(calc), sqrt(1 - calc))
        pubbook.distance_km = 6373 * val
        if pubbook.distance_km <= account.distance:

published_instances_result.append(pubbook)
        published_instances_result.sort(key = lambda x: -
x.distance_km)
        return render(request,
"booshare/publishedbook_instances.html", {"book":
publishedbook_get, "instances": published_instances_result})

def find_books_view(request):
    writer_name = self.request.GET.get('writer', None)
    book_title = self.request.GET.get('book', None)
    country_name = self.request.GET.get('country', None)
    book_genres = self.request.GET.get('genres', None)
    book_tags = self.request.GET.get('tags', None)
    publisher = self.request.GET.get('publisher', None)
    book_language = self.request.GET.get('language', None)
    book_type = self.request.GET.get('type', None)
    publication_year = self.request.GET.get('yeareq', None)
    publication_year_start =
self.request.GET.get('yearstr', None)
    publication_year_end = self.request.GET.get('yearend',
None)

    books = []
    if writer_name:
        writer = Writer.objects.filter(full_name__icontains
= writer_name)
        if writer:
            books = Book.objects.filter(writers = writer)
            if not books:
                return render(request,
"booshare/find_books.html", {"books": []})
        if book_title and not books:
            books = Book.objects.filter(full_title__icontains =
book_title)
            if not books:

```

```

        return render(request,
"booshare/find_books.html", {"books": []})
    elif book_title and books:
        books = books.filter(full_title__icontains =
book_title)
        if not books:
            return render(request,
"booshare/find_books.html", {"books": []})
    if country_name and not books:
        writers = Writer.objects.filter(living_country =
country_name)
        books = Book.objects.filter(writers__in = writers)
        if not books:
            return render(request,
"booshare/find_books.html", {"books": []})
    elif country_name and books:
        writers = Writer.objects.filter(living_country =
country_name)
        books = books.filter(writers__in = writers)
        if not books:
            return render(request,
"booshare/find_books.html", {"books": []})
    if book_genres:
        genres_list = []
        genres_analysis = book_genres.split(',')
        for gen in genres_analysis:
            genre = Genre.objects.get(name = gen)
            genres_list.append(genre)
    if genres_list and books:
        books = books.filter(genres = genres_list)
        if not books:
            return render(request,
"booshare/find_books.html", {"books": []})
    elif genres_list and not books:
        books = Book.objects.filter(genres = genres_list)
        if not books:
            return render(request,
"booshare/find_books.html", {"books": []})
    if book_tags:
        tags_list = []
        tags_analysis = book_tags.split(',')
        for t in tags_analysis:
            tag = Tag.objects.get(name = t)
            tags_list.append(tag)
    if tags_list and books:
        books = books.filter(tags = tags_list)
        if not books:

```

```

        return render(request,
"booshare/find_books.html", {"books": []})
    elif tags_list and not books:
        books = Book.objects.filter(tags = tags_list)
        if not books:
            return render(request,
"booshare/find_books.html", {"books": []})
        publishedbooks = []
        if books:
            books = books.order_by("-full_title")
            publishedbooks =
Publishedbook.objects.filter(book__in = books)
            if publisher and not publishedbooks:
                publishers = Publisher.objects.filter(name =
publisher)
                publishedbooks =
Publishedbook.objects.filter(publisher__in = publishers)
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})
            elif publisher and publishedbooks:
                publishers = Publisher.objects.filter(name =
publisher)
                publishedbooks =
publishedbooks.filter(publisher__in = publishers)
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})
            if book_language and publishedbooks:
                lang = Language.objects.get(name = book_language)
                publishedbooks = publishedbooks.filter(languages =
lang)
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})
            elif book_language and not publishedbooks:
                lang = Language.objects.get(name = book_language)
                publishedbooks =
Publishedbook.objects.filter(languages = lang)
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})
            if book_type and publishedbooks:
                publishedbooks = publishedbooks.filter(book_type =
int(book_type))
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})

```

```

        elif book_type and not publishedbooks:
            publishedbooks =
Publishedbook.objects.filter(book_type = int(book_type))
            if not publishedbooks:
                return render(request,
"booshare/find_books.html", {"books": []})
            if publication_year and publishedbooks:
                publishedbooks =
publishedbooks.filter(first_published_year =
int(publication_year))
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})
            elif publication_year and not publishedbooks:
                publishedbooks =
Publishedbook.objects.filter(first_published_year =
int(publication_year))
                if not publishedbooks:
                    return render(request,
"booshare/find_books.html", {"books": []})
                if publishedbooks:
                    publishedbooks = publishedbooks.order_by("-
average_critique ")
                    return render(request, "booshare/find_books.html",
{"books": books, "publishedbooks": publishedbooks})
                else:
                    return render(request, "booshare/find_books.html",
{"books": books, "publishedbooks": []})

import datetime

def find_book_instances_view(request):
    book_instances = []
    account = Account.objects.get(id = request.user.id)
    account_distance = account.distance
    address_from = Accountaddress.objects.filter(account =
account, base = True)[0]
    book_instances = Bookinstance.objects.filter(approved =
True, shared =
True).prefetch_related("publishedbook__book__writers").prefetch_
related("publishedbook__book__genres").prefetch_related("publish
edbook__book__tags").prefetch_related("addresses").prefetch_rela
ted("sharingrules").select_related("owner_account")
    book_condition = self.request.GET.get('condit', None)
    average_review_from = self.request.GET.get('lreview',
None)
    average_review_to = self.request.GET.get('rreview',
None)

```

```

buy_possibility = self.request.GET.get('buy', None)
no_deposit = self.request.GET.get('deposit', None)
shared_options = self.request.GET.get('share', None)
delivery_possibility = self.request.GET.get('deliv',
None)

book_condition_from = self.request.GET.get('lcondit',
None)

book_condition_to = self.request.GET.get('rcondit',
None)

writer_name = self.request.GET.get('writer', None)
book_title = self.request.GET.get('book', None)
country_name = self.request.GET.get('country', None)
book_genres = self.request.GET.get('genres', None)
book_tags = self.request.GET.get('tags', None)
publisher = self.request.GET.get('publisher', None)
book_language = self.request.GET.get('language', None)
book_type = self.request.GET.get('type', None)
publication_year = self.request.GET.get('yeareq', None)
publication_year_start = self.request.GET.get('lyear',
None)

publication_year_end = self.request.GET.get('ryear',
None)

share_date_from = self.request.GET.get('lsharedate',
None)

share_date_end = self.request.GET.get('rsharedate',
None)

deposit_from = self.request.GET.get('ldeposit', None)
deposit_end = self.request.GET.get('rdeposit', None)
pay_from = self.request.GET.get('lpay', None)
pay_end = self.request.GET.get('rpay', None)
if book_condition:
    book_instances =
book_instances.filter(condition__gte = int(book_condition))
else:
    if book_condition_from:
        book_instances =
book_instances.filter(condition__gte = int(book_condition_from))
    if book_condition_to:
        book_instances =
book_instances.filter(condition__lte = int(book_condition_to))
    if average_review_from:
        book_instances =
book_instances.filter(average_review__gte =
float(average_review_from))
    if average_review_to:
        book_instances =
book_instances.filter(average_review__lte =
float(average_review_to))

```

```

        if buy_possibility:
            book_instances = book_instances.filter(buy_option =
True)
        if no_deposit:
            book_instances =
book_instances.filter(deposit_amount = 0)
            share_options = shared_options.split(',')
            share_list = []
            for share in share_options:
                share_list.append(int(share))
            if share_list:
                book_instances =
book_instances.filter(shared_option__in = share_list)
            if delivery_possibility:
                book_instances = book_instances.filter(delivered =
True)
            if writer_name:
                writer = Writer.objects.filter(full_name__icontains
= writer_name)
                if writer:
                    book_instances =
book_instances.filter(publishedbook__book__writers = writer)
                if book_title:
                    book_instances =
book_instances.filter(publishedbook__book__full_title__icontains
= book_title)
                if country_name:
                    writers = Writer.objects.filter(living_country =
country_name)
                    book_instances =
book_instances.filter(publishedbook__book__writers__in =
writers)
            if book_genres:
                genres_list = []
                genres_analysis = book_genres.split(',')
                for gen in genres_analysis:
                    genre = Genre.objects.get(name = gen)
                    genres_list.append(genre)
            if genres_list:
                book_instances =
book_instances.filter(publishedbook__book__genres = genres_list)
            if book_tags:
                tag_list = []
                tag_analysis = book_tags.split(',')
                for t in tag_analysis:
                    tag = Tag.objects.get(name = gen)
                    tag_list.append(tag)
            if tag_list:

```

```

        book_instances =
book_instances.filter(publishedbook__book__tags = tag_list)
        if publisher:
            publishers = Publisher.objects.filter(name =
publisher)
            book_instances =
book_instances.filter(publishedbook__publisher__in = publishers)
            if book_language:
                lang = Language.objects.get(name = book_language)
                book_instances =
book_instances.filter(publishedbook__languages = lang)
            if book_type:
                book_instances =
book_instances.filter(publishedbook__book_type = int(book_type))
            if publication_year:
                book_instances =
book_instances.filter(published_year = int(publication_year))
            else:
                if publication_year_start:
                    book_instances =
book_instances.filter(published_year__gte =
int(publication_year_start))
                    if publication_year_end:
                        book_instances =
book_instances.filter(published_year__lte =
int(publication_year_end))
                    book_instances_result = []
                    for inst in book_instances:
                        bookaddress = instaddresses.order_by("-
when_added")[0]
                        delta_long = address_from.address_long -
bookaddress.accountaddress.address_long
                        delta_latd = address_from.address_latd -
bookaddress.accountaddress.address_latd
                        calc = sin(delta_latd / 2)**2 +
cos(bookaddress.accountaddress.address_latd) *
cos(address_from.address_latd) * sin(delta_long / 2)**2
                        val = 2 * atan2(sqrt(calc), sqrt(1 - calc))
                        distance_km = 6373 * val
                        if distance_km <= account.distance:
                            book_instances_result.append(inst)
                    if share_date_from:
                        date_elements = share_date_from.split('.')
                        day = int(date_elements[0])
                        month = int(date_elements[1])
                        year = int(date_elements[2])
                        share_date_from = datetime.datetime(year, month,
day)

```

```

    if share_date_to:
        date_elements = share_date_to.split('.')
        day = int(date_elements[0])
        month = int(date_elements[1])
        year = int(date_elements[2])
        share_date_to = datetime.datetime(year, month, day)
    if share_date_from:
        book_instances_result_temp = book_instances_result
        book_instances_result = []
        future_starts = datetime.now()
        for inst in book_instances_result_temp:
            was_ordered = True
            if not share_date_to:
                orders =
Shareorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__lte = share_date_from, when_received__lte =
share_date_to)
                if not orders:
                    orders =
Changeorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__lte = share_date_from, when_received__lte =
share_date_to)
                if not orders:
                    was_ordered = False
            else:
                orders =
Shareorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__gte = share_date_from, to_return__lte =
share_date_to)
                if not orders:
                    orders =
Shareorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__lte = share_date_from, to_return__gte =
share_date_from, to_return__lte = share_date_to)
                if not orders:
                    orders =
Shareorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__gte = share_date_from, when_received__lte =
share_date_to, to_return__gte = share_date_to)
                if not orders:
                    orders =
Shareorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__lte = share_date_from, to_return__gte =
share_date_to)
                if not orders:
                    orders =
Changeorder.objects.filter(bookinstances = inst, stage__gte = 1,

```

```

when_received__gte = share_date_from, to_return__lte =
share_date_to)

                                if not orders:
                                    orders =

Changeorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__lte = share_date_from, to_return__gte =
share_date_from, to_return__lte = share_date_to)
                                if not orders:
                                    orders =

Changeorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__gte = share_date_from, when_received__lte =
share_date_to, to_return__gte = share_date_to)
                                if not orders:
                                    orders =

Changeorder.objects.filter(bookinstances = inst, stage__gte = 1,
when_received__lte = share_date_from, to_return__gte =
share_date_to)

                                if not orders:
                                    was_ordered

= False

                                if not was_ordered:
                                    orders =

Openshareorder.objects.filter(stage__gte = 1, bookinstances =
inst, when_received__gte = future_starts)
                                if orders:
                                    was_ordered = True
                                else:
                                    orders =

Buyorder.objects.filter(stage__gte = 1, bookinstance = inst,
when_received__gte = future_starts)
                                if orders:
                                    was_ordered = True
                                if not was_ordered:
                                    if (0 in share_list or 3 in share_list) and
share_date_to:

                                        period = share_date_to share_date_from
                                        period_days = period.days
                                        sharingrules =
inst.sharingrules.filter(actual = True)
                                        ready_sharingrule_saved = None
                                        for rule in sharingrules:
                                            sharingrule = rule.sharingrule
                                            if sharingrule.max_days <=
period_days:

                                                if not ready_sharingrule_saved:
                                                    ready_sharingrule_saved =
sharingrule

                                                else:

```

```

if sharingrule.max_days >
ready_sharingrule_saved.max_days:
ready_sharingrule_saved
= sharingrule
inst.time_deposit =
ready_sharingrule_saved.deposit
inst.time_price =
ready_sharingrule_saved.pay_amount
to_add = True
if deposit_from:
if inst.time_deposit <
int(deposit_from):
to_add = False
if deposit_to:
if inst.time_deposit > int(deposit_to):
to_add = False
if pay_from:
if inst.time_price < int(pay_from):
to_add = False
if pay_to:
if inst.time_price > int(pay_to):
to_add = False
if to_add:
book_instances_result.append(inst)
book_instances_result.sort(key = lambda x: -
x.average_review)
return render(request,
"booshare/find_book_instances.html", {"books":
book_instances_result})

def writer_view(request, id = None):
writer = None
books = []
if id:
writer = Writer.objects.get(pk =
id).select_related('living_country').select_related('born_city')
.prefetch_related('writers')
books = Book.objects.filter(writers =
writer).order_by('-first_published')[:4]
return render(request, "booshare/writer.html",
{"writer": writer, "books": books})

def publishedbook_view(request, pub_id = None):
publishedbook_get = None
book_instances = []
critiques = []
account = Account.objects.get(id = request.user.id)

```

```

        address_from = Accountaddress.objects.filter(account =
account, base = True)[0]
        if pub_id:
            publishedbook_get =
Publishedbook.objects.prefetch_related("languages").select_relat
ed("book").get(pk = pub_id, None)
            if publishedbook_get:
                critiques =
Critique.objects.filter(publishedbook =
publishedbook_get).order_by("-when_created")[:4]
                book_instances =
Bookinstance.objects.filter(publishedbook = publishedbook_get)
                for inst in book_instances:
                    bookaddress =
BookinstanceAddress.objects.filter(bookinstance =
inst).order_by("-when_added")[0]
                    delta_long = address_from.address_long -
bookaddress.accountaddress.address_long
                    delta_latd = address_from.address_latd -
bookaddress.accountaddress.address_latd
                    calc = sin(delta_latd / 2)**2 +
cos(bookaddress.accountaddress.address_latd) *
cos(address_from.address_latd) * sin(delta_long / 2)**2
                    val = 2 * atan2(sqrt(calc), sqrt(1 - calc))
                    inst.distance_km = 6373 * val
                    book_instances = book_instances.order_by("-
distance_km")[:4]
                return render(request, "booshare/publishedbook.html",
{"book": publishedbook_get, "instances": book_instances,
"critiques": critiques})

    def country_writers_view(request, country_id = None):
        country_writers = []
        countries = Country.objects.all().order_by("name")
        if country_id:
            country_get = Country.objects.get(pk =
int(country_id), None)
            if country_get:
                country_writers =
Writer.objects.filter(Q(living_country = country_get) |
Q(born_city__country = country_get)).order_by('full_name')
                return render(request,
"booshare/country_writers.html", {"writers": country_writers,
"countries": countries})
            return render(request, "booshare/country_writers.html",
{"writers": [], "countries": countries})

    def city_writers_view(request, city_id = None):

```

```

        city_writers = []
        countries = Country.objects.all().order_by("name")
        country_cities = {}
        for country in countries:
            country_cities[country.name] =
City.objects.filter(country = country)
            if city_id:
                city_get = City.objects.get(pk = int(city_id),
None)

                if city_get:
                    city_writers = Writer.objects.filter(born_city
= city_get).order_by('full_name')
                    return render(request,
"booshare/city_writers.html", {"writers": city_writers,
"country_cities": country_cities})
                return render(request, "booshare/city_writers.html",
{"writers": [], "country_cities": country_cities})

def publisher_books_view(request, publisher_id = None):
    publisher_books = []
    if publisher_id:
        publisher_get = Publisher.objects.get(pk =
publisher_id, None)
        if publisher_get:
            publisher_books =
Publishedbook.objects.filter(publisher =
publisher_get).prefetch_related("languages")
            select_related("book").order_by('-average_critique')
            return render(request, "booshare/publisher_books.html",
{"books": publisher_books})

import uuid
import settings

def add_book_view(request):
    check = request.user.is_active
    if check:
        if request.method == "POST":
            book_instance = BookinstanceForm(request.POST)
            if book_instance.is_valid():
                book_instance_result =
book_instance.save(commit=False)
                book_instance_result.owner_account =
Account.objects.get(id = request.user.id)
                book_instance_result.approved = True
                book_instance.longcode =
uuid.uuid5(uuid.NAMESPACE_DNS, settings.addr +
str(book_instance.id))

```

```

        book_instance_result.save()
        book_instance.save_m2m()
        redirect("ibookview", book_instance.id)
    else:
        book_instance = BookinstanceForm()
    else:
        book_instance = None
    return render(request, "booshare/my_book.html",
{"form": book_instance })

def edit_book_view(request, id):
    check = request.user.is_active
    book_instance = Bookinstance.objects.get(pk = id, None)
    if check and book_instance and
book_instance.owner_account == Account.objects.get(id =
request.user.id):
        if request.method == "POST":
            book_instance = BookinstanceForm(request.POST,
instance = book_instance)
            if book_instance.is_valid():
                book_instance.save()
                book_instance.save_m2m()
                return redirect("ibookview",
book_instance.id)
            else:
                book_instance = BookinstanceForm(None, instance
= book_instance)
            else:
                book_instance = None
        return render(request, "booshare/my_book.html",
{"form": book_instance })

def add_new_book_instance_view(request):
    check = request.user.is_active
    if check:
        if request.method == "POST":
            book = BookForm(request.POST)
            published_book = PublishedForm(request.POST)
            book_instance = BookinstanceForm(request.POST)
            if book_instance.is_valid() and
published_book.is_valid() and book_instance.is_valid():
                book_res = book.save(commit = False)
                book_res.approved = False
                book_res.save()
                book.save_m2m()
                published_book_res =
published_book.save(commit = False)
                published_book_res.approved = False

```

```

        published_book_res.save()
        published_book.save_m2m()
        book_instance_result =
book_instance.save(commit=False)
        book_instance_result.owner_account =
Account.objects.get(id = request.user.id)
        book_instance_result.approved = False
        book_instance_result.save()
        book_instance.save_m2m()
        redirect("ibookview", book_instance.id)
    else:
        book = BookForm()
        published_book = PublishedForm()
        book_instance = BookinstanceForm()
    else:
        book_instance = None
        published_book = None
        book = None
    return render(request, "booshare/add_my_book.html",
{"book_form": book, "published_book": published_book,
"instance_form": book_instance })

def add_book_application_view(request, id):
    check = request.user.is_active
    if check:
        if request.method == "POST":
            book_application =
BookinstanceApplicationForm(request.POST)
            if book_application.is_valid():
                book_application_result =
book_application.save(commit=False)
                if id:
                    book_instance =
Bookinstance.objects.get(pk = id, None)
                    if book_instance:
                        book_application_result.bookinstance =
book_instance
                        book_application_result.owner_account =
Account.objects.get(id = request.user.id)
                        book_instance_result.approved = False
                        book_instance_result.save()
                        redirect("ibookview", book_instance.id)
                else:
                    book_application =
BookinstanceApplicationForm()
            else:
                book_application = None

```

```

        return render(request,
"booshare/book_application.html", {"form": book_application})

def add_sharingrule_view(request):
    check = request.user.is_active
    if check:
        if request.method == "POST":
            sharing_rule = SharingruleForm(request.POST)
            if sharing_rule.is_valid():
                sharing_rule_result =
sharing_rule.save(commit=False)
                sharing_rule_result.account =
Account.objects.get(id = request.user.id)
                sharing_rule_result.save()
                redirect("sharingrulesview")
        else:
            sharing_rule = SharingruleForm()
    else:
        sharing_rule = None
    return render(request, "booshare/sharing_rule.html",
{"form": sharing_rule})

def edit_sharingrule_view (request, id):
    check = request.user.is_active
    sharingrule = Sharingrule.objects.get(pk = id, None)
    if check and sharingrule and sharingrule.account ==
Account.objects.get(id = request.user.id):
        if request.method == "POST":
            sharing_rule = SharingruleForm(request.POST,
instance = sharingrule)
            if sharing_rule.is_valid():
                sharing_rule.save()
                return redirect("sharingrulesview")
        else:
            sharing_rule = SharingruleForm(None, instance =
sharingrule)
    else:
        sharing_rule = None
    return render(request, "booshare/sharing_rule.html",
{"form": sharing_rule})

def sharingrules_view(request):
    account = Account.objects.get(id = request.user.id)
    if account.is_active:
        account_sharingrules =
Sharingrule.objects.filter(account =
account).prefetch_related("bookinstances").order_by('-id')
    else:

```

```

        account_sharingrules = []
        return render(request, "booshare/my_sharingrules.html",
{"sharingrules": account_sharingrules })

def book_instances_view(request):
    account = Account.objects.get(id = request.user.id)
    if account.is_active:
        account_bookinstances =
Bookinstance.objects.filter(owner_account =
account).prefetch_related("sharerules").select_related("publishe
dbook").select_related("publishedbook__book").prefetch_related("
publishedbook__book__writers").order_by('-id')
    else:
        account_bookinstances = []
        return render(request, "booshare/my_books.html",
{"bookinstances": account_bookinstances})

def book_instance_view(request, book_id = None):
    book_get = None
    sharingrules = []
    reviews = []
    book = None
    publishedbook = None
    if book_id:
        book_get = Bookinstance.objects.get(pk = book_id,
None)

        if book_get:
            publishedbook = Publishedbook.objects.get(pk =
book_get.publishedbook.id)
            book = Book.objects.get(pk =
publishedbook.book.id)
            sharingrules =
BookinstanceSharingrule.objects.filter(bookinstance = book_get,
actual = True).select_related("sharingrule")
            reviews =
BookinstanceReview.objects.filter(bookinstance =
book_get).order_by("-when_created")
            reviews_count = reviews.count()
        return render(request, "booshare/book_inst.html",
{"book_instance": book_get, "book": book, "published_book":
publishedbook, "sharingrules": sharingrules, "reviews":
reviews[:4], "reviews_count": reviews_count})

```

Б.2 Текст файла `critique_views.py`

```

from .models import Critique, Account, Publishedbook,
Publishedbok, AccountReview, BookinstanceReview
from .forms import CritiqueForm, BookinstanceReviewForm,
AccountReviewForm

def publishedbook_critiques_view(request, pub_id = None):
    publishedbook_id = None
    published_critiques = []
    if publishedbook_id:
        publishedbook_get =
Publishedbook.objects.select_related("book").get(pk =
publishedbook_id, None)
        if publishedbook_get:
            published_critiques =
Critique.objects.filter(publishedbook =
publishedbook_get).order_by("-when_created")
            return render(request, "booshare/pub_critiques.html",
{"publishedbook": publishedbook_get, "critiques":
published_critiques})

def account_critiques_view(request, account_id = None):
    account_critiques = {}
    if account_id:
        account_get = Account.objects.get(pk = account_id,
None)
        if account_get:
            publishedbooks =
Critique.objects.filter(account = account_get).order_by("-
when_created").distinct("publishedbook")
            for publish in publishedbooks:
                account_critiques[publish] =
Critique.objects.filter(account = account_get, publishedbook =
publish).order_by("-when_created")
            else:
                account_get = Account.objects.get(id =
request.user.id)
                publishedbooks = Critique.objects.filter(account =
account_get).order_by("-when_created").distinct("publishedbook")
                for publish in publishedbooks:
                    account_critiques[publish] =
Critique.objects.filter(account = account_get, publishedbook =
publish).order_by("-when_created")
                return render(request,
"booshare/account_critiques.html", {"name": account.first_name,

```

```

"last_name": account.last_name, "critiques": account_critiques
}))

def add_critique_view(request, pub_id = None):
    check = request.user.is_active
    if pub_id:
        publishedbook = Publishedbok.objects.get(pk =
pub_id, None)
    if check and publishedbook:
        if request.method == "POST":
            critique = CritiqueForm(request.POST)
            if critique.is_valid():
                critique_result = critique.save(commit =
False)
                critique_result.publishedbook =
publishedbook
                critique_result.account =
Account.objects.get(id = request.user.id)
                critique_result.save()
                redirect("accountcritiques")
            else:
                critique = CritiqueForm()
        else:
            critique = None
    return render(request, "booshare/critique.html",
{"publishedbook": publishedbook, "form": critique})

def account_reviews_view(request, account_id = None):
    account_reviews = []
    bookinstance_reviews = []
    if account_id:
        account_get = Account.objects.get(pk = account_id,
None)
        if account_get:
            account_reviews =
AccountReview.objects.filter(account_author =
account_get).order_by("-when_created")
            bookinstance_reviews =
BookinstanceReview.objects.filter(account =
account_get).order_by("-when_created")
        else:
            account_get = Account.objects.get(id =
request.user.id)
            account_reviews =
AccountReview.objects.filter(account_author =
account_get).order_by("-when_created")

```

```

        bookinstance_reviews =
BookinstanceReview.objects.filter(account =
account_get).order_by("-when_created")
        return render(request, "booshare/account_reviews.html",
{"name": account.first_name, "last_name": account.last_name,
"reviews": account_reviews, "book_reviews": bookinstance_reviews
})

def bookinstance_reviews_view(request, book_id = None):
    bookinstance_reviews = []
    if book_id:
        book_get = Bookinstance.objects.get(pk = book_id,
None)

        if book_get:
            bookinstance_reviews =
BookinstanceReview.objects.filter(bookinstance =
book_get).order_by("-when_created")
            return render(request,
"booshare/bookinstance_reviews.html", {"book_instance":
book_get, "reviews": bookinstance_reviews})

def shared_reviews_view(request, account_id = None):
    shared_reviews = []
    if account_id:
        account_get = Account.objects.get(pk = account_id,
None)

        if account_get:
            shared_reviews =
AccountReview.objects.filter(account_share =
account_get).order_by("-when_created")
        else:
            account_get = Account.objects.get(id =
request.user.id)
            shared_reviews =
AccountReview.objects.filter(account_share =
account_get).order_by("-when_created")
            return render(request, "booshare/account_reviews.html",
{"name": account.first_name, "last_name": account.last_name,
"reviews": shared_reviews})

def add_book_review_view(request, book_id = None):
    check = request.user.is_active
    allowed = False
    if book_id:
        book_instance = Bookinstance.objects.get(pk =
book_id, None)
        account = Account.objects.get(id = request.user.id)

```

```

        orders = Shareorder.objects.filter(bookinstances =
book_instance, account = account, stage__gte = 2)
        if orders:
            allowed = True
        else:
            orders =
Changeorder.objects.filter(bookinstances = book_instance,
account = account, stage__gte = 3)
            if orders:
                allowed = True
            else:
                orders =
Changeorder.objects.filter(bookinstances = book_instance,
to_account = account, stage__gte = 3)
                if orders:
                    allowed = True
                else:
                    orders =
Openshareorder.objects.filter(bookinstances = book_instance,
account = account, stage__gte = 2)
                    if orders:
                        allowed = True
                    else:
                        orders =
Buyorder.objects.filter(bookinstance = book_instance, account =
account, stage__gte = 2)
                        if orders:
                            allowed = True
    if check and book_instance and allowed:
        if request.method == "POST":
            book_inst =
BookinstanceReviewForm(request.POST)
            if book_inst.is_valid():
                book_inst_result = book_inst.save(commit =
False)

                book_inst_result.bookinstance = book_inst
                book_inst_result.account = account
                book_inst_result.save()
                redirect("accountcritiques")
            else:
                book_inst = BookinstanceReviewForm()
        else:
            book_inst = None
    return render(request, "booshare/review.html", {"book":
book_instance, "form": book_inst})

def add_account_review_view(request, acc_id = None):
    check = request.user.is_active

```

```

        if acc_id:
            account_about = Account.objects.get(pk = acc_id,
None)
            books = Bookinstance.objects.filter(owner_account =
account_about)
            account = Account.objects.get(id = request.user.id)
            orders =
Shareorder.objects.filter(bookinstances__in = books, account =
account, stage__gte = 2)
            if orders:
                allowed = True
            else:
                orders =
Changeorder.objects.filter(bookinstances__in = books, account =
account, stage__gte = 3)
            if orders:
                allowed = True
            else:
                orders =
Changeorder.objects.filter(bookinstances__in = books, to_account
= account, stage__gte = 3)
            if orders:
                allowed = True
            else:
                orders =
Openshareorder.objects.filter(bookinstances__in = books, account
= account, stage__gte = 2)
            if orders:
                allowed = True
            else:
                orders =
Buyorder.objects.filter(bookinstance__in = book, account =
account, stage__gte = 2)
            if orders:
                allowed = True
        if check and account_about and allowed:
            if request.method == "POST":
                acc_review = AccountReviewForm(request.POST)
                if acc_review.is_valid():
                    acc_review_result = acc_review.save(commit
= False)
                    acc_review_result.account_share =
account_about
                    acc_review_result.account_author =
Account.objects.get(id = request.user.id)
                    acc_review_result.save()
                    redirect("accountcritiques")
            else:

```

```
        acc_review = AccountReviewForm()
    else:
        acc_review = None
    return render(request, "booshare/account_review.html",
{"account": account_about, "form": acc_review})
```

ДОДАТОК В
Слайди презентації

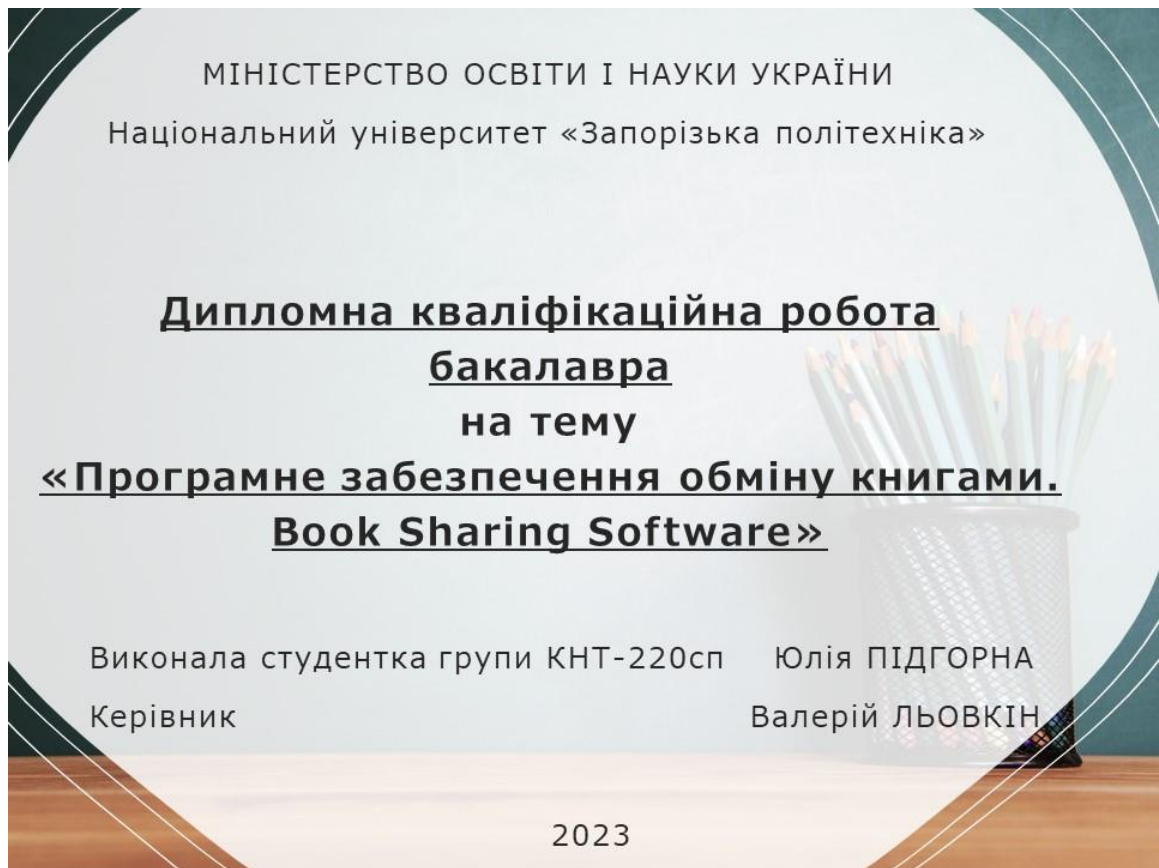


Рисунок В.1 – Початковий слайд

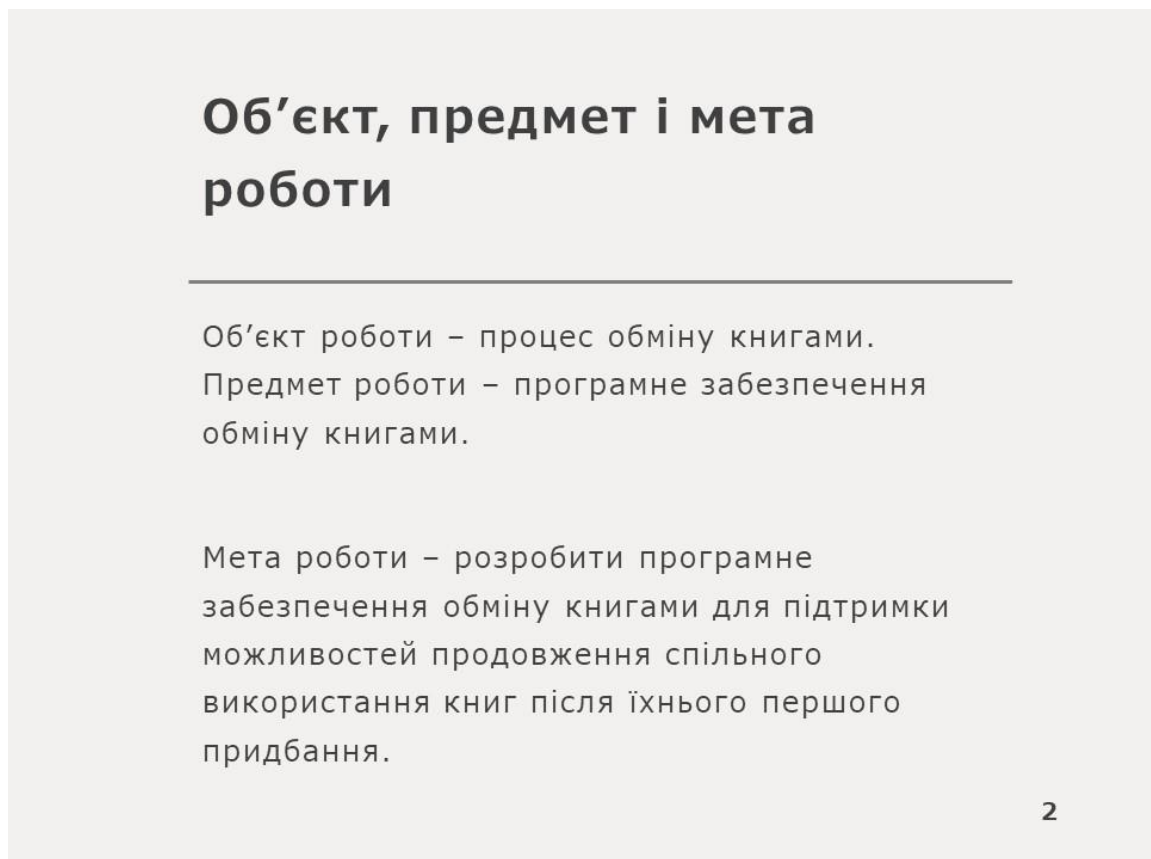


Рисунок В.2 – Мета дипломної кваліфікаційної роботи

Порівняння аналогів з розробкою

Характеристика	GoodReads	Little Free Library	Share A Book	Розробка
Рецензії	Книги	–	Коментарі в Facebook	Видання, екземпляри книг
Обмін книгами	–	Без сплати	Планується без сплати	Зі сплатою, без сплати, з заставою
Замовлення книг для обміну	–	–	–	Так
Придбання книг, що були у використанні	–	–	Так	Так

3

Рисунок В.3 – Порівняння аналогів з розробкою

Порівняння вебфреймворків для розробки

Показник	Django	Pyramid
Підтримка Bootstrap	+	+
Архітектурний патерн	Model-View-Controller	Model-View-Controller
Система підтримки шаблонів	+	Можливо підключити
Робота зі структурою додатку	Можливість створювати декілька додатків у одному проєкті	Один проєкт як один додаток
Інтерфейс адміністратора	+	–
Підтримка високого трафіку на практиці	+	+
Підтримка щодо документації	++	+

4

Рисунок В.4 – Порівняння вебфреймворків для розробки

Діаграма прецедентів

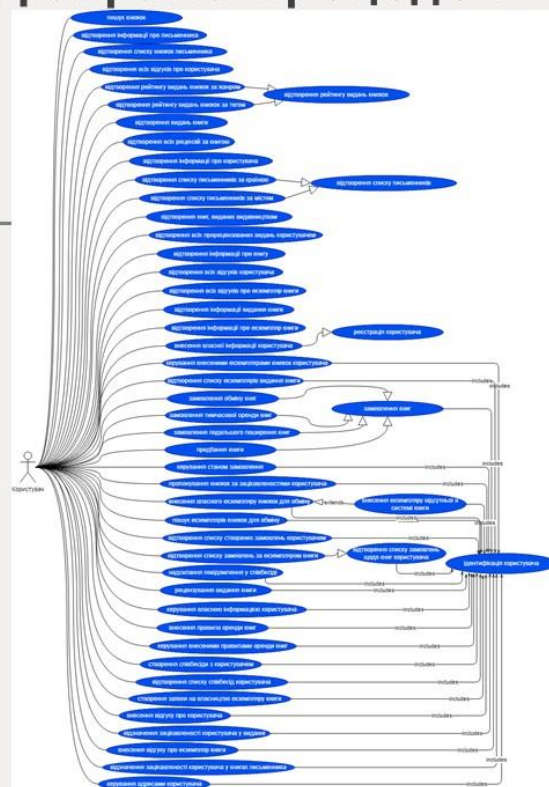


Рисунок В.5 – Діаграма прецедентів

Схема бази даних



Рисунок В.6 – Схема бази даних



Рисунок В.7 – Діаграми станів замовлення подальшого поширення книг та замовлення обміну книг



Рисунок В.8 – Діаграма станів замовлення тимчасової оренди книг

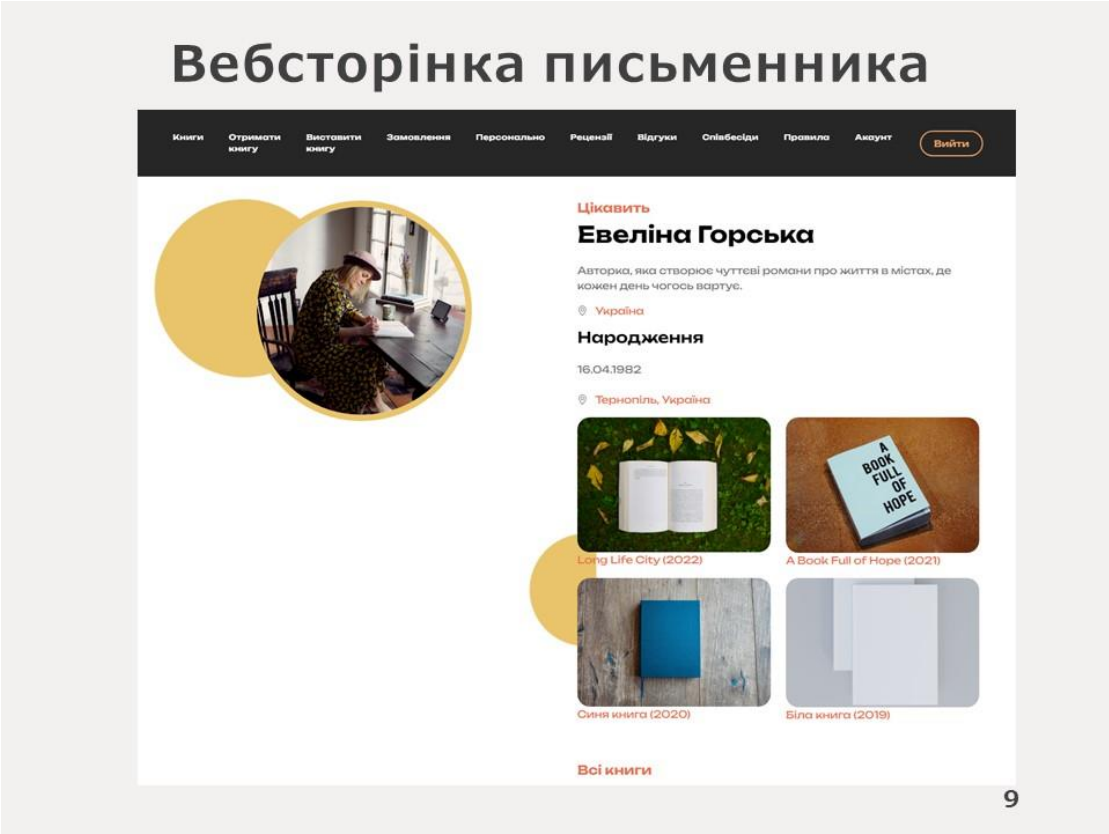


Рисунок В.9 – Вебсторінка письменника

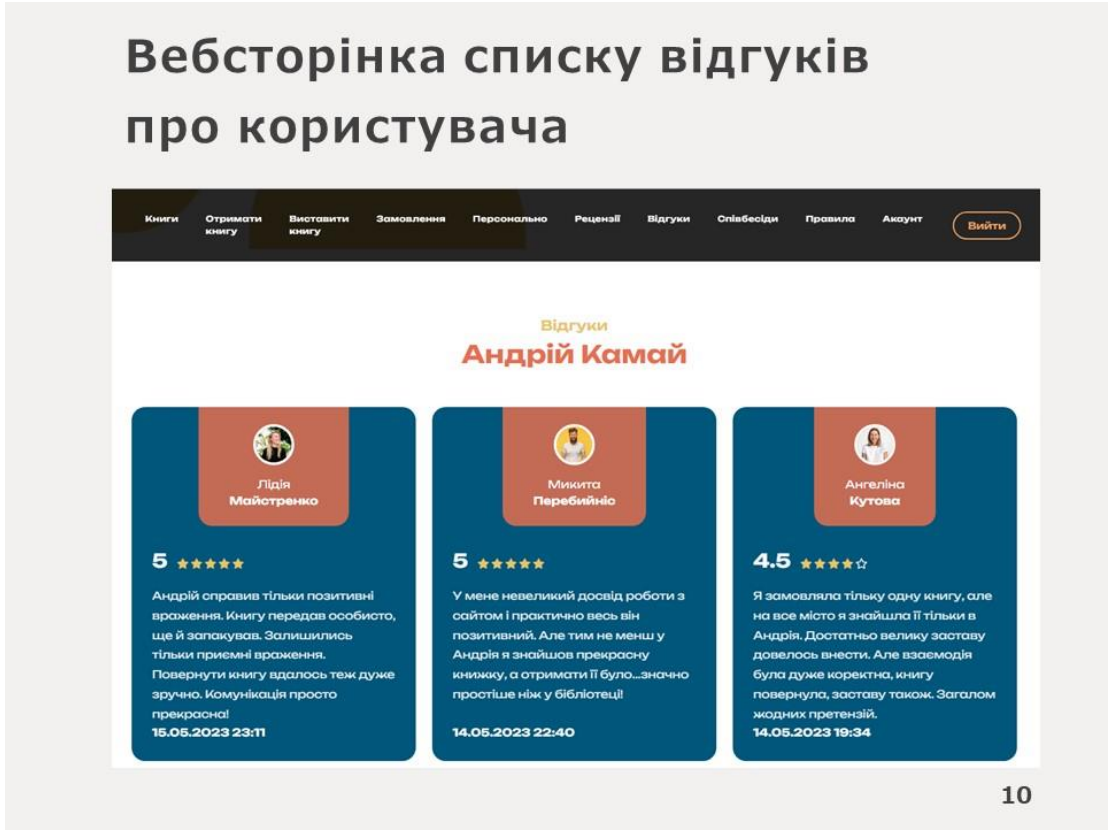
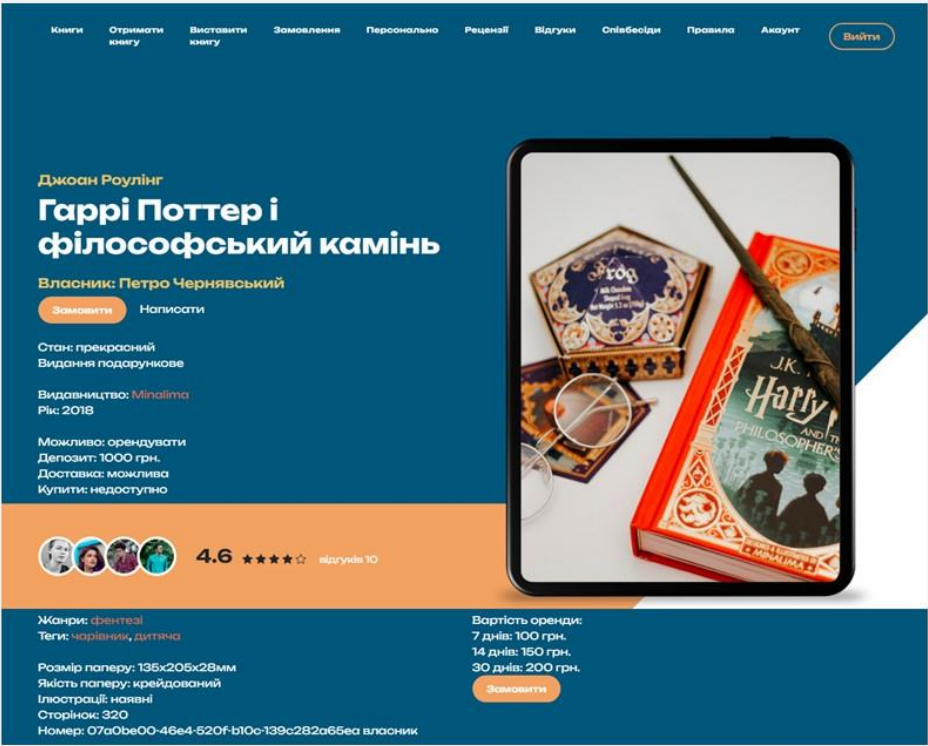


Рисунок В.10 – Вебсторінка списку відгуків про користувача

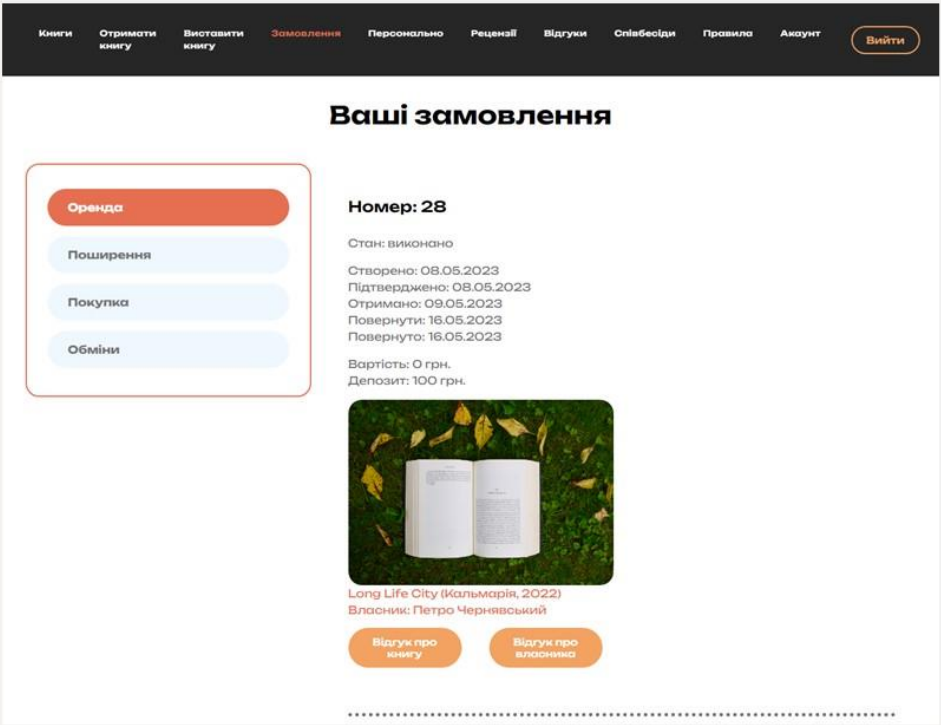
Вебсторінка екземпляру книги



11

Рисунок В.11 – Вебсторінка екземпляру книги

Вебсторінка власних замовлень



12

Рисунок В.12 – Вебсторінка власних замовлень

Результати тестування програми

Сценарій	Підсумок	Сценарій	Підсумок
Пошук екземплярів книжок для обміну	Успішно	Відтворення всіх рецензій за книгою	Успішно
Пошук книжок	Успішно	Відтворення всіх прорецензованих видань користувачем	Успішно
Пропонування книжок за зацікавленостями користувача	Успішно	Відтворення всіх відгуків користувача	Успішно
Внесення власного екземпляру книжки для обміну	Успішно	Відтворення всіх відгуків про користувача	Успішно
Керування внесеними екземплярами книжок користувача	Успішно	Відтворення всіх відгуків про екземпляр книги	Успішно
Внесення екземпляру відсутньої в системі книги	Успішно	Відтворення інформації про книгу	Успішно
Відтворення списку замовлень за екземпляром книги	Успішно	Відтворення інформації видання книги	Успішно
Відтворення списку екземплярів видання книги	Успішно	Відтворення інформації про екземпляр книги	Успішно
Замовлення тимчасової оренди книг	Успішно	Створення заявки на власництво екземпляру книги	Успішно
Замовлення подальшого поширення книг	Успішно	Створення співбесіди з користувачем	Успішно
Замовлення обміну книг	Успішно	Надсилення повідомлення у співбесіді	Успішно
Керування станом замовлення	Успішно	Відтворення списку співбесід користувача	Успішно
Відтворення списку створених замовлень користувачем	Успішно	Внесення правила оренди книг	Успішно
Відтворення списку замовлень щодо книг користувача	Успішно	Керування внесеними правилами оренди книг	Успішно
Придбання книги	Успішно	Рецензування видання книги	Успішно
Внесення власної інформації користувача	Успішно	Внесення відгуку про екземпляр книги	Успішно
Реєстрація користувача	Успішно	Внесення відгуку про користувача	Успішно
Керування власною інформацією користувача	Успішно	Відтворення списку письменників за країною	Успішно
Керування адресами користувача	Успішно	Відтворення списку письменників за містом	Успішно
Відтворення інформації про письменника	Успішно	Відтворення книг, виданих видавництвом	Успішно
Відтворення інформації про користувача	Успішно	Відзначення зацікавленості користувача у виданні	Успішно
Відтворення рейтингу видань книжок за жанром	Успішно	Відзначення зацікавленості користувача у книгах письменника	Успішно
Відтворення рейтингу видань книжок за тегом	Успішно		
Відтворення видань книги	Успішно		

Рисунок В.13 – Результати тестування програми

Висновки

Визначено структуру бази даних програми з виділенням сутностей книги, видання книги, екземпляра книги, різних варіантів замовлень, та відповідні функції і сценарії роботи користувача з цими функціями. Програму реалізовано мовою Python за допомогою засобів веброзробки, включаючи HTML, CSS, вебфреймворк Django, та системи MySQL для роботи з базою даних. Усі реалізовані функції за визначеними для програми сценаріями представлені у вигляді програмних функцій у складі комбінації уявлення і шаблонів за структурою вебфреймворку Django.

Створена програма надає можливість не тільки передати тимчасово книгу в користування іншій людині, але і фактично підтримувати розширення її подальшого використання за рахунок страхування інтересів власника на основі внесення застави, за рахунок надання гнучкої системи варіантів обміну та надання інформації про орендарів і конкретні екземпляри книг, а також конкретні видання книг на основі використання рецензування і відгуків.

Реалізовану програму було описано з представленням основної структури роботи користувача з програмою, деталізованим виділенням основних функціональних можливостей, а функціональні можливості було відтестовано з доведенням до успішних результатів за кожним з запланованих сценаріїв.

Рисунок В.14 – Висновки за дипломною кваліфікаційною роботою