НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Михайло ШВИДЕНКО** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Веб-орієнтована інформаційна система обліку книжкових ресурсів між користувачами»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми _____ **Роман РУДЕНСЬКИЙ**
д.т.н., доцент
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Ярослав ПОНЗЕЛЬ**
Доктор філософії з комп. наук
(підпис)

Консультант _____ **Тетяна ВОЛОШИНА**
Доцент, канд. пед. наук
(підпис)

Виконала _____ **Анастасія ФАСТОВЕЦЬ**
(підпис)

Київ - 2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інформаційних систем і технологій

к.е.н., доцент Михайло ШВИДЕНКО

(підпис) (ПІБ)

“_06_” _січня_ 2026 р.

ЗАВДАННЯ

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ЗДОБУВАЧУ

Фастовець Анастасії Валеріївни

(прізвище, ім’я, по батькові)

Спеціальність «122 Комп’ютерні науки»
Освітньо-професійна програма «Комп’ютерні науки» Тема бакалаврської кваліфікаційної роботи

«Веб-орієнтована інформаційна система обліку книжкових ресурсів між користувачами» затверджена наказом від «06» січня 2026 р. №46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи:

Вимоги до автоматизації процесу обліку та обміну книгами; необхідність створення централізованого каталогу літератури користувачів; стек технологій: мова програмування Python, фреймворк Django, СУБД PostgreSQL.

Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області та аналогів систем буккросингу.
2. Проєктування архітектури системи та інтеграція із зовнішніми сервісами книжкових даних (Google Books API).

3. Розробка програмного забезпечення з використанням елементів машинного навчання для рекомендацій.

4. Тестування функціоналу, впровадження системи та аналіз ефективності роботи алгоритмів.

Дата видачі завдання «06» січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ **Ярослав ПОНЗЕЛЬ**
(підпис)
Доктор філософії з комп. наук

Консультант _____ **Тетяна ВОЛОШИНА**
Доцент, канд. пед. наук
(підпис)

Завдання взяла до виконання _____ **Анастасія ФАСТОВЕЦЬ**
(підпис)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ВІС – веб-орієнтована інформаційна система

ІС – інформаційна система

ПЗ – програмне забезпечення

СУБД – система управління базами даних

API – прикладний програмний інтерфейс

CSS – каскадні таблиці стилів

EHI – індекс життєздатності екосистеми

HTML – мова розмітки гіпертексту

HTTP/HTTPS – протокол передання гіпертексту / безпечний протокол передання гіпертексту

KPI – ключовий показник ефективності

MVT – архітектурний шаблон проєктування.

ORM – об'єктно-реляційне відображення

UI/UX – користувацький інтерфейс та користувацький досвід

UML – уніфікована мова моделювання

TF-IDF – Term Frequency-Inverse Document Frequency

ISBN – міжнародний стандартний книжковий номер

ВСТУП

У сучасному цифровому світі, з урахування постійного розвиток електронних та аудіоформатів, паперова книга все ж залишається важливим інструментом інтелектуального та культурного розвитку особистості. Читання завжди було і лишається основним елемент критичного мислення та соціалізації. Проте динаміка на книжкового ринку показує невтішні результати, а джерела вказують, що наразі помітне суттєве зростання вартості і нових, і старих видань, що робить всю літературу менш доступною для більшої частини верств населення. Такі проблеми однозначно породжують потребу в альтернативних способах розповсюдження інформації, зокрема через реалізацію вже вживаної книги.

Нинішня ситуація на книжковому ринку показує невизначеність читача, адже насправді існує величезна кількість приватних бібліотек, але концепція такого читання книги часто є непривабливою для споживача. Так як попит на ті самі книги паралельно все ж присутній, можна дійти висновку, що потрібні нові методи збуту вторинної книги, який буде зручний для людини, максимально автоматизований. Традиційні методи обміну літературою зазвичай обмежені фізичними локаціями: бібліотеками, книгарнями або локальними «полицями книжкового обміну» в кав’ярнях. Така модель має низьку ефективність через територіальну обмеженість, відсутність переліку доступних видань та неможливість оперативного пошуку потрібного примірника.

Аналіз існуючих рішень показує, що більшість онлайн-платформ для книгообміну є або застарілими, або вузькоспеціалізованими. Соціальні мережі та платформи з оголошеннями не забезпечують необхідного рівня автоматизації обліку та безпеки. Часто такі сервіси не мають централізованої системи перевірки стану книг, механізмів відстеження обмінів або додаткових ініціатив, наприклад, благодійного спрямування. В результаті, потенціал книжкової

спільноти залишається нереалізованим, а процес дарування чи обміну книгами перетворюється на хаотичний і складний процес.

Створення веб-орієнтованої інформаційної системи на базі сучасних фреймворків дозволяє створити процес взаємодії між книголюбами геть на новому рівні. Автоматизація обліку книжкових ресурсів дасть змогу систематизувати особисті бібліотеки споживачів, а також створити глобальну мережу доступної літератури. Це має стати екосистемою, де користувач може легко знайти книгу, запропонувати обмін, отримати видання в дарунок або взяти участь у благодійних зборах. Використання технологій веброзробки забезпечує доступність сервісу в Україні, а також закордоном, де налаштована інтеграція Нової Пошти.

Метою кваліфікаційної роботи є проєктування архітектури та програмна реалізація сучасної веб-орієнтованої інформаційної системи для обліку, обміну та благодійного розповсюдження книжкових ресурсів. Ця система покликана створити зручне і безпечне цифрове середовище для ефективної взаємодії читачів, забезпечити високу прозорість процесу обміну літературою, а також сприяти популяризації читання та раціональному використанню вже надрукованих видань в умовах постійного зростання їхньої вартості на ринку.

Для досягнення поставленої мети було сформовано комплекс науково-практичних завдань. Насамперед передбачається проведення детального аналізу предметної області, дослідження існуючих платформ для книгообміну та виявлення їхніх ключових недоліків. Наступним кроком є проєктування архітектури майбутнього рішення, розробка оптимізованої реляційної бази даних та створення ергономічного користувацького інтерфейсу. Найважливішим завданням є безпосередня програмна реалізація системи з подальшою інтеграцією зовнішніх програмних інтерфейсів. Завершальним етапом стане комплексне тестування, профілювання та оптимізація вебзастосунка для забезпечення його стабільної роботи при високих навантаженнях.

Практичне значення одержаних результатів полягає у створенні функціонально завершеного програмного продукту, який вирішує реальну

суспільну проблему доступності літератури. Розроблений продукт може бути успішно впроваджений як самостійний соціальний або благодійний проєкт, що об'єднує читачів не лише в межах України, а й за її межами. Впровадження такої системи дозволить значно скоротити час на пошук потрібних видань, автоматизувати логістичні процеси та надати книгам друге життя, підтримуючи ідеї екологічного споживання та розширення культурного простору.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність теми та проблематика обміну книг

Сучасний стан книжкового ринку України та світу характеризується стрімкою трансформацією. Попри активний перехід на цифрові технології, паперова книга залишається як джерелом інформації, так й об’єктом колекціонування, естетичного задоволення та важливим інструментом соціальної комунікації. Проте доступність друкованої книги стикається з низкою економічних та логістичних бар’єрів, що робить розробку систем автоматизованого обміну книжковими ресурсами критично необхідною.

Одним із головних чинників, що обмежують доступ до літератури, є її вартість. Згідно з даними Українського інституту книги та провідних ритейлерів, таких як «Книгарня Є» чи «Yakaboo», середня вартість якісного видання в твердій обкладинці у 2023–2024 роках коливається в межах 350–600 грн. Для багатьох верств населення, зокрема студентства та мешканців невеликих міст, систематичне оновлення власної бібліотеки стає фінансово обтяжливим. Як зазначав відомий письменник Ніл Ґейман у своїх лекціях про читання: «Ми маємо обов'язок читати для задоволення... якщо ми читаємо, ми змінюємо світ». Проте в умовах, коли купівля 2-3 книг на місяць складає значну частку прожиткового мінімуму, «зміна світу» стає розкішшю.

Використання системи обміну дозволяє перетворити «заморожений капітал» у вигляді прочитаних книг на активний ресурс. Користувач, витративши кошти один раз, отримує потенційний доступ до десятків інших видань через механізм рециркуляції.

Також, виробництво паперових книг – це ресурсомісткий процес. Для виготовлення однієї тонни паперу потрібно близько 24 дерев, тисячі літрів води та значна кількість електроенергії, вже не кажучи, що логістика та поліграфічні

фарби мають свій вуглецевий слід. Концепція свідомого читання передбачає багаторазове використання одного примірника різними людьми. Книга, що роками стоїть на полиці після одного прочитання, з точки зору екології – нелогічний ресурс. Система обміну сприяє подовженню життєвого циклу книги, зменшуючи потребу у перевиробництві та стимулюючи відповідальне споживання, а також в такій ініціативі є великий культурний потенціал.

Наразі книжковий обмін в Україні реалізований фрагментарно. Точки обміну часто знаходять в мережевих або локальних книгарнях, для цього виділяють окремі полиці. В таких місцях література часто буде або неактуальною, або застарілою. Більш зручний варіант – це обмін в соціальних мережах, спеціалізованих каналах зв'язку за інтересами. Але за такого варіанту неможливо забезпечити безпекою подібну операцію, важким також є пошук і загалом весь життєвий цикл такої події.

Розробка веб-орієнтованої інформаційної системи орієнтована на наступну цільову аудиторію:

- Студентську молодь для обміну навчальною та фаховою літературою.
- Батьків, оскільки дитячі книги швидко втрачають актуальність через дорослішання дитини.
- Колекціонерів та поціновувачів рідкісних видань, яким потрібен інструмент для пошуку специфічних примірників.
- Волонтерські організації, які можуть просунути волонтерські збори, за рахунок книжкового обміну.

Відсутність централізованої платформи для обміну призводить до ряду негативних наслідків. Мешканці населених пунктів, де немає книгарень, залишаються без доступу до нових знань. Це призводить до занепаду культури читання, складність і дорожнеча доступу до книг змушують користувачів переходити на електронний формат, іноді навіть звертаються до піратського цифрового контенту, або взагалі відмовлятися від читання на користь швидкого черпання інформації з соцмереж.

Отже, актуальність розробки зумовлена необхідністю створення зручного, безпечного та технологічного інструменту, який об’єднає особисті бібліотеки в єдину інформаційну мережу. Використання сучасного стеку технологій дозволить забезпечити високу швидкість обробки даних та масштабованість системи, що є критичним для формування загальнонаціональної спільноти книгообміну.

1.2 Аналіз існуючих інформаційних систем у сфері книжкового обміну

Сучасний ринок цифрових послуг пропонує обмежену кількість спеціалізованих платформ для обміну фізичними книгами, що змушує користувачів використовувати адаптовані рішення або локальні спільноти. Найбільш релевантним та технологічно досконалим прикладом на українському ринку є платформа «Liberbee». Дана система позиціонує себе як повноцінний маркетплейс та майданчик для книжкового обміну, надаючи користувачам можливість не лише обмінюватися літературою, а й здійснювати її купівлю та продаж. Важливою перевагою «Liberbee» є інтеграція комерційної складової, що дозволяє залучати ширшу аудиторію та забезпечувати фінансову стабільність ресурсу. Система має розвинений інтерфейс каталогізації, що значно спрощує пошук видань за жанрами чи авторами. Попри високий рівень реалізації, сервіс орієнтований переважно на модель «маркетплейсу», де соціальні аспекти обміну та благодійні ініціативи часто відходять на другий план перед комерційними транзакціями.

Іншим сегментом існуючих рішень є глобальні соціальні мережі для книголюбів, такі як «Goodreads» або український аналог «Rork». Ці системи володіють потужним інструментарієм для трекінгу прочитаного та формування особистих списків, проте вони майже повністю позбавлені функціоналу для прямого фізичного обміну між користувачами. У таких системах обмін реалізується лише через сторонні обговорення у коментарях або групах, що не

забезпечує безпеки та автоматизації процесу. Користувач змушений самостійно домовлятися про умови передачі, логістику та верифікацію стану книги, що підвищує поріг входження та ризики для учасників.

Також варто відзначити спеціалізовані західні платформи, такі як «BookMooch» або «PaperbackSwap», що базуються на системі балів. У таких моделях користувач отримує бали за відправлену книгу, які потім може витратити на замовлення іншої літератури. Попри ефективність моделі обміну з наданням балів користувачу за різні активності, ці системи часто мають застарілий користувацький інтерфейс та не адаптовані до українських реалій логістики. Також вони не передбачають інтеграції з сучасними платіжними системами та не підтримують локальні благодійні програми, що є актуальним для сучасного українського реалізму.

Майбутня система має усунути недоліки конкурентів шляхом впровадження прозорих механізмів дарування та обміну, з акцентом на автоматизацію обліку та підтримку волонтерських ініціатив, що зробить її більш привабливою для некомерційних спільнот та академічного середовища.

1.3 Визначення цілей та завдань дослідження

Головною метою даної бакалаврської роботи є проєктування та програмна реалізація повнофункціональної веб-орієнтованої інформаційної системи, призначеної для автоматизації процесів обліку, пошуку та безпечного обміну книжковими ресурсами між користувачами. Ця система має стати єдиним централізованим середовищем, яке не лише спрощує логістику вторинного обігу літератури, але й формує надійну соціальну інфраструктуру для читацької спільноти.

Досягнення поставленої мети вимагає глибокого аналізу користувацького досвіду та впровадження сучасних принципів ергономіки програмного забезпечення. Фундаментальною концепцією розробки є мінімізація когнітивного навантаження на кінцевого споживача. Інформаційна архітектура

та навігаційні потоки проєктуються таким чином, щоб користувач міг отримати максимально релевантний та якісний результат за мінімально можливу кількість ітерацій. Інтуїтивна зрозумілість функціоналу виступає критичним фактором успішності продукту, адже цільова аудиторія платформи охоплює людей із різним рівнем цифрової грамотності.

Водночас, за зовнішньою простотою клієнтського інтерфейсу повинна безперебійно функціонувати складна багаторівнева система верифікації та обробки даних. Щоб гарантувати високу якість фінального результату, програмний комплекс налаштовується на самостійне виконання всього спектра фонових перевірок. Це включає валідацію введених користувачем параметрів, запобігання дублюванню записів у базі даних, автоматичний контроль цілісності реляційних зв'язків при зміні статусів угод, а також предиктивний моніторинг аномалій через аналітичні модулі.

Для досягнення сформульованої мети та реалізації описаної концепції в межах дослідження поставлено такі ключові завдання:

- здійснити системний аналіз предметної області, виявити архітектурні недоліки існуючих рішень та сформувати деталізовані вимоги до створюваного програмного забезпечення;
- розробити логічну та фізичну структуру бази даних, здатну ефективно обробляти транзакційні запити та зберігати складні зв'язки між сутностями системи без втрати швидкодії;
- спроєктувати архітектуру вебзастосунка з використанням сучасних шаблонів проєктування, чітко відокремивши бізнес-логіку, роботу з масивами даних та представлення інформації;
- програмно реалізувати механізми управління життєвим циклом книгообміну, включаючи логіку зміни статусів, систему тригерних сповіщень та інструментарій для модерації користувацьких скарг;
- імплементувати математичний аналітичний модуль для автоматизованого моніторингу працездатності платформи;

- розробити адаптивний користувацький інтерфейс, що забезпечує безперешкодний доступ до функціоналу платформи з різних типів пристроїв при збереженні високих стандартів юзабіліті;
- провести комплексне тестування створеної інформаційної системи для перевірки коректності роботи алгоритмів, стійкості до відмов та відповідності початковим специфікаціям.

Вирішення вищезазначених завдань у своїй сукупності забезпечить створення не просто технічного каталогу, а повноцінного інструменту автоматизації бізнес-процесів. Результатом роботи стане масштабована, безпечна та зручна у використанні система, яка бере на себе всі складні обчислення та перевірки, залишаючи користувачу лише чистий та інтуїтивний досвід обміну ресурсами.

1.4 Формулювання вимог до інформаційної системи «BookSharing»

Процес розробки надійного програмного забезпечення невіддільний від етапу чіткого визначення вимог. Як зазначається у фаховій літературі з програмної інженерії, повнота та несуперечливість сформованих вимог на початкових етапах життєвого циклу проєкту дозволяє мінімізувати архітектурні ризики та суттєво знизити витрати часу на подальшому етапі. З огляду на специфіку веб-орієнтованої інформаційної системи обміну книгами, комплекс вимог доцільно розділити на дві фундаментальні категорії: функціональні та нефункціональні. Цей поділ базується на загальноприйнятих індустріальних стандартах проєктування архітектури додатків імплементації [15, с. 31].

Функціональні вимоги визначають базову поведінку системи, її реакції на вхідні дані та вичерпний перелік операцій, доступних для різних категорій користувачів. У контексті методології проєктування важливо враховувати, що «функціональні вимоги до програмного забезпечення описують сервіси, які мають надаватися цим забезпеченням, те, як програмне забезпечення має

реагувати на конкретні вхідні дані, та те, як воно має поводитися в певних ситуаціях» [11, с. 10]. Керуючись цим фундаментальним принципом, для інформаційної системи «BookSharing» було деталізовано логіку реалізації сервісів та підтримувану обробку запитів для трьох основних ролей: неавторизованого відвідувача, авторизованого учасника платформи та модератора.

Авторизований користувач виступає ключовим актором екосистеми. До функціональних вимог цієї ролі належать: можливість генерування, редагування та видалення власних оголошень про наявні для обміну примірники; формування формалізованих запитів на отримання літератури від інших учасників; управління життєвим циклом транзакції шляхом зміни її статусів у базі даних. Окрім цього, система повинна надавати механізм подання скарг у разі порушення умов обміну іншою стороною, що є критично важливим інструментом для підтримки високого рівня довіри всередині читацької спільноти.

Функціональні вимоги до підсистеми адміністрування охоплюють управління обліковими записами потенційних порушників за допомогою системи накопичувальних попереджень, ручну модерацію сумнівного контенту та доступ до розширеного аналітичного дашборду. Система має автоматично розраховувати та візуалізувати на панелі модератора поточні значення індексу життєздатності екосистеми, коефіцієнти успішності угод та рівень загальної ліквідності книжкового фонду.

Якщо функціональні специфікації описують безпосередні можливості користувачів на платформі, то нефункціональні вимоги відповідають за її загальну стабільність та технічну досконалість. Вони охоплюють широкий спектр архітектурних обмежень, починаючи від часу відгуку сервера на клієнтські запити і закінчуючи механізмами шифрування даних. Завдяки чіткому формулюванню цих критеріїв забезпечується високий рівень відмовостійкості, що дозволяє вебзастосунку працювати безпечно та ефективно навіть за умов

стрімкого збільшення кількості активних сесій або тимчасових затримок з боку сторонніх API.

У контексті кібербезпеки та захисту персональних даних, інформаційна система повинна забезпечувати надійне одностороннє хешування паролів користувачів із використанням стійких криптографічних алгоритмів. Архітектура вебзастосунка повинна містити вбудовані механізми захисту від поширених вразливостей, зокрема міжсайтового скриптингу, SQL-ін'єкцій та підробки міжсайтових запитів. Доступ до маршрутів, що вимагають авторизації, має бути суворо обмежений та верифікований на рівні серверної бізнес-логіки.

Вимоги до продуктивності та швидкодії передбачають оптимізацію реляційних запитів до системи управління базами даних PostgreSQL. Час генерації відповіді сервером не повинен перевищувати стандартних показників. Для досягнення цього структура бази даних повинна підтримувати правильне використання зовнішніх ключів та індексів для таблиць, до яких найчастіше виконуються складні запити.

Щодо ергономіки та зручності взаємодії, клієнтська частина платформи має бути реалізована з дотриманням принципів адаптивного вебдизайну. Користувацький інтерфейс повинен зберігати повну структурну цілісність та функціональність незалежно від роздільної здатності екрана пристрою клієнта.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір архітектурного підходу та технологічного стеку

Проєктування будь-якого надійного програмного продукту розпочинається з фундаментального етапу визначення його глобальної архітектури та підбору оптимального набору інструментальних засобів. Для реалізації платформи вторинного обігу літератури було прийнято рішення базуватися на класичній клієнт-серверній архітектурі з використанням патерну проєктування, що забезпечує чітке розмежування бізнес-логіки, рівня зберігання даних та користувацького інтерфейсу. У якості базового архітектурного шаблону обрано підхід Model-View-Template, який є специфічною та високоефективною реалізацією парадигми Model-View-Controller. Цей шаблон дозволяє повністю ізолювати фізичну структуру бази даних від механізмів обробки HTTP-запитів та візуального представлення контенту, що критично важливо для подальшого масштабування програмного коду. Відповідно до цієї концепції, сутності предметної області інкапсулюються в об'єктних моделях, логіка маршрутизації та валідації зосереджується у представленнях, а генерація фінального гіпертекстового документа делегується потужній системі шаблонізації. Такий розподіл відповідальності значно спрощує етап тестування окремих компонентів екосистеми та дозволяє паралельно виконувати завдання з проєктування інтерфейсу та написання серверних алгоритмів обробки транзакцій.

Вибір базової мови програмування та серверного фреймворку був зумовлений жорсткими вимогами щодо швидкості ітеративної розробки, високого рівня вбудованої безпеки та наявності професійного інструментарію для роботи з реляційними масивами даних. Фундаментом back-end частини виступає високорівнева мова програмування Python, яка відзначається лаконічним синтаксисом, високою читабельністю коду та багатою екосистемою аналітичних бібліотек. Роль основного вебфреймворку виконує Django.

Використання саме цього інструменту аргументується його філософією комплексного підходу, що передбачає наявність готових модулів для вирішення типових завдань інженерії без необхідності глибокої інтеграції сторонніх рішень. Зокрема, фреймворк надає розширену систему об'єктно-реляційного відображення, яка повністю абстрагує розробника від написання низькорівневих запитів базовою мовою маніпулювання даними, дозволяючи оперувати записами як класичними об'єктами. Крім того, інструментарій містить готову адміністративну панель, що кардинально прискорює процес створення закритого інтерфейсу для модераторів платформи, та надійні механізми захисту від найпоширеніших критичних вебвразливостей.

Збереження абсолютної консистентності даних є першочерговим завданням для систем, що керують взаємними транзакціями між користувачами мережі. З огляду на це, для організації рівня зберігання інформації обрано систему управління реляційними базами даних PostgreSQL. На відміну від нереляційних рішень, ця СУБД забезпечує повну та безкомпромісну підтримку вимог стандарту ACID, гарантуючи атомарність, узгодженість, ізольованість та довговічність виконання кожної операції обміну книгами. Враховуючи складну архітектуру предметної області книжкового обміну де профілі учасників, унікальні примірники книг, заявки на транзакції та рейтингові оцінки тісно пов'язані через систему зовнішніх ключів, реляційна модель є найбільш доцільним та безпечним вибором. Важливим фактором на користь даного рішення є також підтримка розширених механізмів індексування, що оптимізує процеси повнотекстового пошуку літератури в глобальному каталозі. Висока продуктивність рушія при виконанні складних з’єднувальних розрахунків є необхідною умовою для коректної та швидкої роботи модуля аналітики при обчисленні композитних індексів стабільності платформи.

Розробка клієнтської частини системи фокусується на забезпеченні максимальної ергономіки та повної адаптивності інтерфейсу на різних типах пристроїв. Візуальне представлення екранів будується за сучасним принципом першочергової орієнтації на мобільні платформи. Для швидкої стандартизації

елементів управління та типографіки застосовується каскадний фреймворк Bootstrap п'ятої версії у тісному поєднанні з класичним набором технологій розмітки браузера. Відмова від використання важковагових клієнтських середовищ на користь швидкої серверної генерації сторінок дозволила суттєво зменшити час початкового завантаження вебзастосунка. Синтез обраних технологічних рішень утворює надійну інфраструктуру, яка повністю задовольняє актуальні потреби автоматизації вторинного обігу літератури та закладає потенціал для подальшого масштабування функціоналу.

2.2 Логічна і фізична модель бази даних

Під час проєктування інформаційно-логічної моделі бази даних платформи «BookSharing» особливу увагу було приділено забезпеченню цілісності даних та усуненню надликовості інформації. Це повністю узгоджується з класичним визначенням, згідно з яким «база даних — це пойменована, структурована сукупність взаємопов'язаних даних, які відображають стан об'єктів та їх взаємозв'язків у розглядуваній предметній області» [12, с. 6].

Розробка була поділена на два етапи: на створення концептуально-логічної моделі, що відображає бізнес-сутність предметної області, та її подальшу трансформація у фізичну модель, адаптовану під специфіку обраної реляційної СУБД PostgreSQL.

Логічна модель даних була спроєктована з суворим дотриманням правил нормалізації до третьої нормальної форми, що дозволило мінімізувати структурну надмірність та запобігти аномаліям модифікації даних. Відповідно до розширених функціональних вимог предметної області соціального обміну книг, ядро системи було розкладено на декілька логічних блоків: управління профілями та логістикою, глибока каталогізація літератури, управління фізичними примірниками з їх медіафайлами, а також облік транзакцій.

Фундаментальною архітектурною особливістю спроєктованої бази даних є концептуальне розділення абстрактного видання та його реального фізичного

носія. Такий підхід дозволив оптимізувати роботу з візуальним контентом: у таблиці абстрактної книги зберігаються посилання на офіційні цифрові обкладинки від видавництв, тоді як таблиця фізичного примірника, прив'язана до конкретного власника, містить шляхи до реальних фотографій книги. Це дає змогу користувачам візуально підтверджувати фактичний стан та ступінь зношеності літератури перед ініціалізацією обміну.

Каталогізація книжкового фонду реалізована через розгалужену систему нормалізованих довідників. Окрім базових таблиць жанрів та авторів, до системи інтегровано сутність видавництв, що підвищує точність бібліографічного опису. Інноваційною складовою інформаційної моделі є таблиця ідентифікатора категорії (tag). Ця сутність була закладена в архітектуру спеціально для накопичення та структурування масивів даних, які в подальшому оброблятимуться алгоритмами машинного навчання з метою кластеризації літератури та формування прогнозованих персоналізованих рекомендацій. Оскільки одне видання може належати до різних жанрів, бути написаним у співавторстві та описуватися множиною тегів, між цими сутностями реалізовано зв'язки типу «багато-до-багатьох» через проміжні асоціативні таблиці.

Окрема увага при проєктуванні приділялася логістичному потенціалу платформи. Для забезпечення безшовного процесу фізичної доставки книг, до моделі користувача було додано додаткові атрибути для збереження конфігураційних даних інтеграції з API поштового оператора «Нова Пошта». Фіксація у профілі користувача прив'язки до конкретного населеного пункту та номера відділення дозволяє в перспективі автоматизувати процес генерації електронних транспортних накладних при успішному підтвердженні угоди.

Підсистема обліку транзакцій представлена таблицею exchange_request, яка агрегує зовнішні ключі на ініціатора запиту та конкретний примірник книги. Для підтримки саморегуляції спільноти та функціонування аналітичних модулів реалізовано таблицю exchange_complaint, що фіксує конфліктні ситуації між учасниками та їх деталі.

Фізична модель бази даних була згенерована та імплементована за допомогою механізмів об'єктно-реляційного відображення серверного фреймворку Django. Забезпечення суворої реляційної цілісності реалізовано на рівні ядра бази даних через декларативні обмеження зовнішніх ключів. Сформована ER-діаграма, яка наочно демонструє всі таблиці бізнес-логіки системи, включаючи логістичні атрибути та підготовлені для машинного навчання сутності (рис. 2.1).

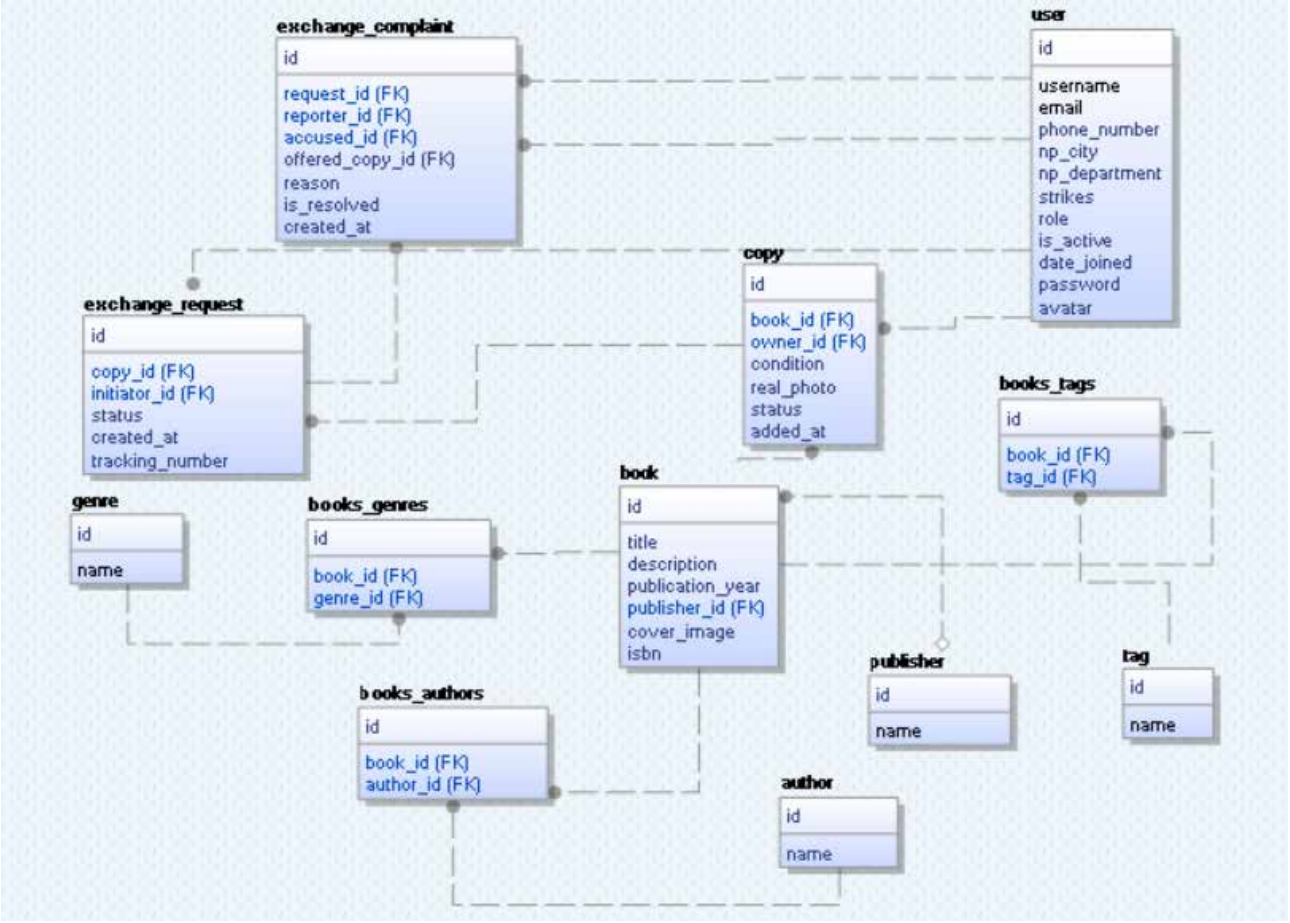

Рис. 2.1 – ER Діаграма інформаційної системи обліку книжкових ресурсів

2.3 Моделювання бізнес-процесів за допомогою UML-діаграм

Для деталізації поведінкових аспектів, візуалізації алгоритмів функціонування та проєктування фізичної інфраструктури інформаційної системи «BookSharing» застосовано методологію об’єктно-орієнтованого моделювання за допомогою Уніфікованої мови моделювання. Цей

індустріальний стандарт дозволяє всебічно та стандартизовано описати архітектуру програмного продукту з різних точок зору, забезпечуючи чітке й однозначне розуміння складних бізнес-процесів. Моделювання екосистеми платформи здійснювалося на концептуальному, логічному та фізичному рівнях, що дозволило системно покрити весь життєвий цикл функціоналу: від первинної взаємодії користувача з інтерфейсом до глибокої обробки даних на back-end та їх розміщення на апаратних вузлах. Для комплексного розкриття прийнятих проєктних рішень архітектура вебзастосунка розглядається через призму чотирьох ключових UML-моделей. Визначення загальних меж системи, специфікація ролей акторів та доступного їм функціоналу здійснюється на етапі аналізу вимог за допомогою діаграми прецедентів. Внутрішня динаміка виконання конкретних бізнес-алгоритмів, зокрема життєвий цикл транзакцій обміну та механізми генерації логістичних даних, ілюструється через діаграму діяльності. Хронологічний порядок обміну повідомленнями між клієнтом, вебсервером та базою даних під час обробки критичних запитів формалізується діаграмою послідовностей. На завершальному етапі просторовий розподіл розроблених програмних модулів на реальних обчислювальних потужностях та конфігурація мережевої взаємодії описуються за допомогою діаграми розгортання. Такий комплексний підхід створює вичерпну візуальну специфікацію, яка слугує надійним інженерним фундаментом для етапу безпосередньої програмної реалізації та подальшого масштабування платформи.

2.3.1 Діаграма прецедентів

Діаграма прецедентів є фундаментальним інструментом для візуального опису функціональних вимог до інформаційної системи та визначення контексту її взаємодії з навколишнім середовищем. Розроблена модель платформи «BookSharing» чітко розмежовує можливості системи залежно від рівня доступу акторів та ілюструє ключові точки інтеграції із зовнішніми сервісами. Архітектура прав доступу побудована з використанням механізму наслідування

навколо базового актора, трьох його спеціалізованих ролей та двох зовнішніх систем, що дозволяє уникнути дублювання зв'язків та покрити весь спектр запланованих бізнес-процесів.

Фундаментом ієрархії виступає базовий актор «Користувач», який володіє загальними правами доступу до відкритих даних платформи, а саме перегляду загального каталогу літератури та використання системи пошуку з фільтрацією. Від нього успадковують свої можливості два інші актори: неавторизований та авторизований користувачі. Унікальною функцією гостя є ініціалізація прецеденту реєстрації та входу в систему. Натомість авторизована дійова особа, яка вже має активну сесію, отримує доступ до широкого спектра закритого функціоналу. Зокрема, цей актор здійснює управління власним профілем, додає фізичні примірники книг до особистої бібліотеки, формує та обробляє запити на обмін літературою, додає ТТН відправлення, а також має право подавати скарги. Вищим рівнем привілеїв у системі володіє модератор. Продовжуючи об'єктно-орієнтований ланцюжок наслідування, цей актор переймає всі повноваження авторизованого користувача, проте його роль розширюється специфічними адміністративними прецедентами. Модератор відповідає за вирішення конфліктів, управління користувачами та загальною статистикою, а також здійснює глобальне управління контентом, що включає редагування або додавання записів про книги.

Важливою архітектурною особливістю спроєктованої моделі є наявність зовнішніх системних акторів, які забезпечують виконання інтеграційних процесів та збагачення даних платформи через відношення включення. Сервіс Google Books виступає зовнішньою системою під час реалізації прецеденту отримання метаданих, що є невіддільною частиною процесу додавання нового примірника книги. Це дозволяє платформі автоматично завантажувати бібліографічну інформацію. Своєю чергою, інтеграція із Сервісом Нова Пошта залучається під час управління профілем користувача. Викликаний прецедент синхронізації міст та відділень забезпечує підготовку актуальної логістичної інфраструктури для організації подальшого фізичного обміну між

користувачами. Такий комплексний розподіл ролей гарантує цілісність даних, надійність саморегуляції спільноти та повністю задовольняє функціональні вимоги до вебзастосунка (рис. 2.2).

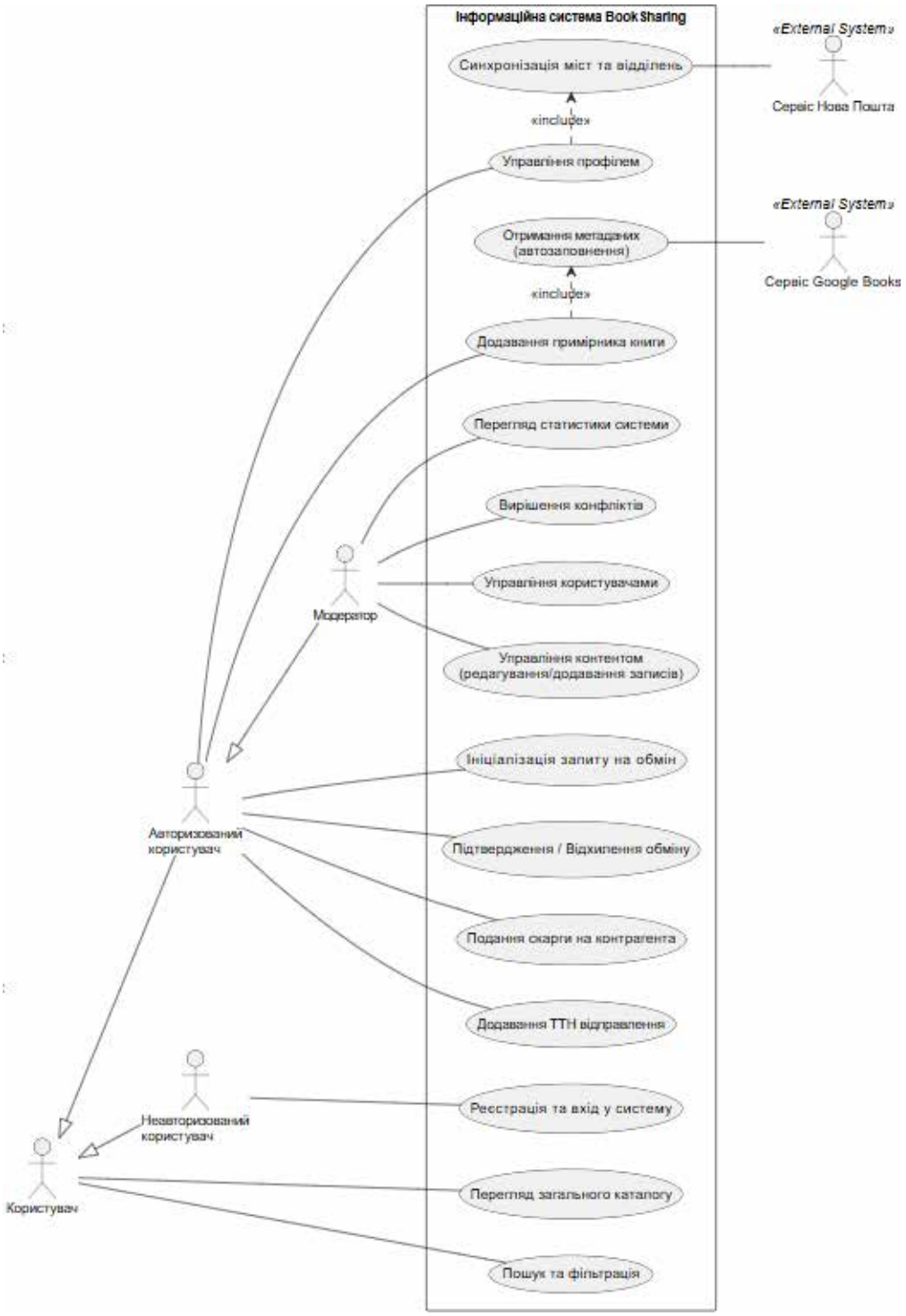

Рис.2.2 – Діаграма прецедентів інформаційної системи обліку книжкових ресурсів

2.3.2 Діаграма діяльності

Діаграма діяльності призначена для моделювання динамічних аспектів поведінки системи, алгоритмів виконання бізнес-процесів та маршрутизації потоків управління. Для інформаційної системи «BookSharing» за допомогою даного типу діаграм було формалізовано ключовий і найбільш багатокроковий процес екосистеми – життєвий цикл транзакції рівноправного двостороннього обміну фізичними примірниками книг. З метою чіткого розмежування зон відповідальності та візуалізації взаємодії між клієнтською і серверною частинами, модель побудована з розподіленням дій між трьома основними суб'єктами: ініціатором обміну, системою та власником книги.

Бізнес-процес розпочинається з дій ініціатора, який за допомогою інструментів пошуку та фільтрації знаходить потрібний примірник у загальному каталозі платформи та активує функцію пропозиції обміну. Отримавши цей тригер, система генерує запит на транзакцію, присвоює йому початковий статус очікування та автоматично направляє відповідне сповіщення власнику цільового примірника. У випадку відхилення пропозиції власником система переводить транзакцію у статус скасованої, надсилає сповіщення ініціатору, і життєвий цикл процесу завершується у системній доріжці. Але за погодження, система фіксує підтвердження, змінює статус запиту на прийнятий та відкриває обом сторонам взаємний доступ до контактних і логістичних даних для оформлення доставки.

Наступний етап алгоритму моделює паралельну асинхронну взаємодію контрагентів, обидва учасники незалежно один від одного здійснюють відправку матеріальних носіїв через відділення «Нової Пошти» та вносять згенеровані номери товарно-транспортних накладних до інтерфейсу системи. Щойно обидва потоки інтегруються у вузлі, система фіксує успішне виконання логістичного етапу та переводить запит у статус «В дорозі».

Після фактичного прибуття відправлень процес знову розгалужується на два паралельні потоки для моделювання процедури отримання та незалежного контролю якості. Кожен із користувачів у своїй зоні відповідальності забирає

посилку та здійснює візуальний огляд отриманої книги. У разі відповідності примірника заявленим характеристикам користувач підтверджує отримання, а в разі виявлення дефектів чи невідповідностей – ініціює формування скарги.

Якщо за результатами обробки відповідей з'ясовується, що обидва учасники задоволені результатом угоди, система остаточно закриває транзакцію зі статусом виконаної. Якщо ж хоча б один із контрагентів сформував скаргу, спрацьовує сценарій обробки виключних ситуацій: система реєструє конфлікт, переводить скаргу у статус «В роботі» та створює робочий тікет для модератора платформи, на чому поточний бізнес-процес обміну вважається вичерпаним. Змодельована діаграма активності детально описує цю логіку (рис. 2.3)

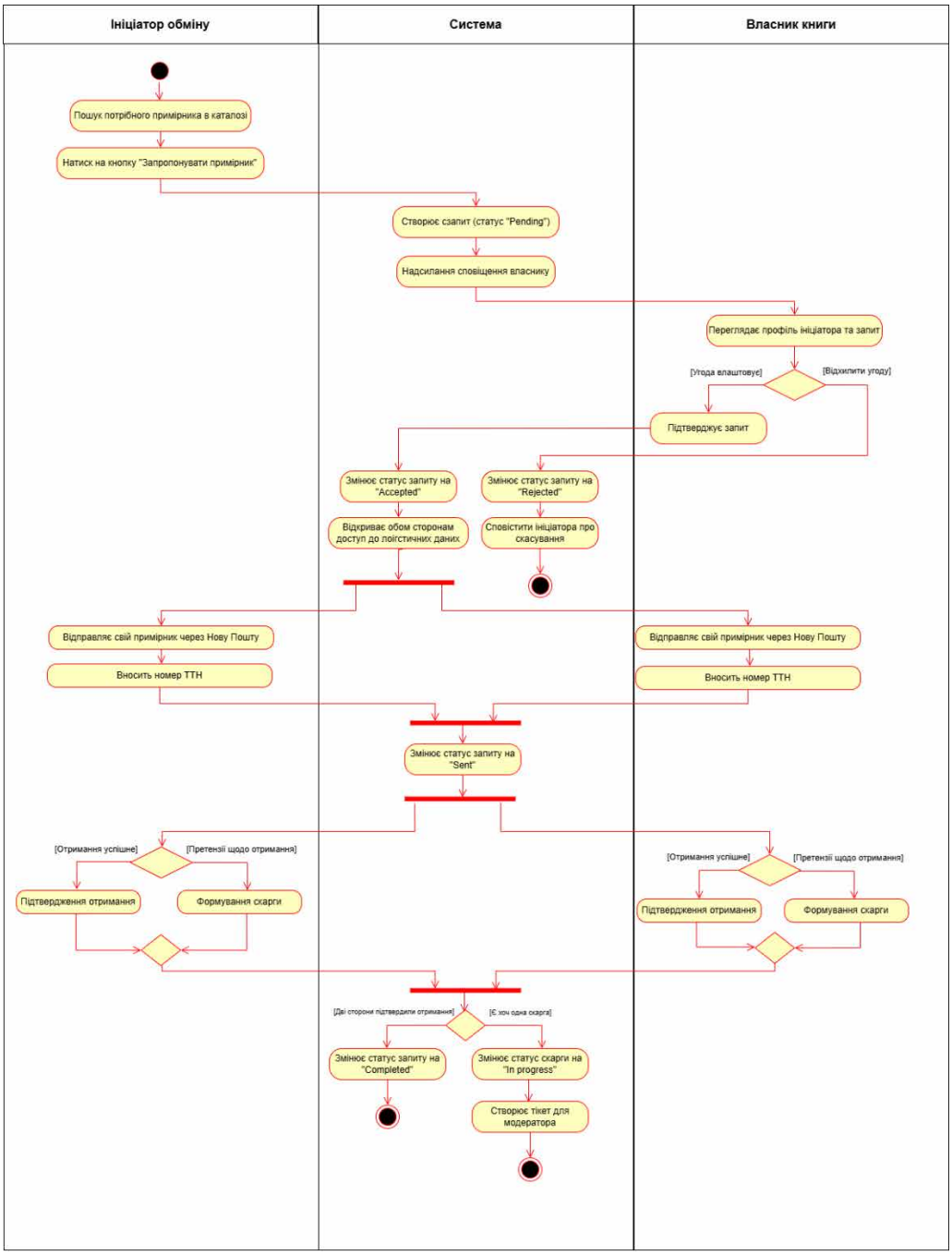

Рис. 2.3 – Діаграма діяльності інформаційної системи обліку книжкових ресурсів

2.3.3 Діаграма послідовності

Для детального моделювання динаміки взаємодії об'єктів у часі та візуалізації хронологічного порядку передачі повідомлень під час виконання ключових бізнес-сценаріїв було розроблено діаграму послідовності. Даний тип

моделі дозволяє розкрити внутрішню логіку роботи системи на рівні програмних викликів, чітко розмежовуючи дії користувачів, обробку даних на сервері, логіку розгалуження та паралельного виконання процесів, а також комунікацію із зовнішніми програмними інтерфейсами. У межах проєктування інформаційної системи «BookSharing» для діаграми послідовності було обрано наскрізний комплексний сценарій, який охоплює повний життєвий цикл обміну книг між двома корситувачами: від етапу первинної реєстрації книги в каталозі до успішного підтвердження транзакції обміну або обробки виключних ситуацій за участю адміністратора платформи. Архітектура моделі побудована навколо п'яти основних ліній життя, що репрезентують ініціатора обміну, власника книги, ядро інформаційної системи «BookSharing», зовнішній сервіс Google Books API та модератора платформи.

Сценарій взаємодії розпочинається з ініціалізації процесу додавання нового примірника власником книги. Отримавши запит, інформаційна система автоматично генерує синхронний виклик до Google Books API для отримання бібліографічних метаданих, таких як автори, жанри, обкладинка. Після повернення структурованої інформації система реєструє об'єкт у базі даних, створює картку оголошення та сигналізує власнику про успішне завершення операції. Наступний етап моделює роботу підсистеми пошуку та рекомендацій: коли ініціатор формує запит на пошук літератури у каталозі, система виконує внутрішню операцію для аналізу користувацьких вподобань та відповідних тегів. Результатом цієї обробки є повернення відфільтрованого списку релевантних видань на клієнтський інтерфейс ініціатора.

Кульмінаційна фаза діаграми описує безпосередньо процес укладання угоди, асинхронної логістичної синхронізації та контролю якості. Процес активується надсиланням запиту на обмін від ініціатора, що призводить до створення запиту та сповіщення власника про нього. Логіка вибору моделюється за допомогою фрейму альтернатив:

У разі відхилення пропозиції власником система оновлює статус запиту на «Rejected» та сповіщає ініціатора, завершуючи життєвий цикл транзакції. А у

разі прийняття пропозиції власником система змінює статус запиту на «Accepted», виконує внутрішню операцію зчитування збережених даних доставки з профілів користувачів та дзеркально відображає логістичні координати Нової Пошти кожній зі сторін угоди.

Наступний етап фізичного відправлення книг моделюється за допомогою фрейму паралельного виконання. Обидва учасники процесу незалежно один від одного здійснюють надсилання посилок та вносять номери товарно-транспортних накладних до інтерфейсу системи. Щойно обидві дії фіксуються, система оновлює статус запиту на «Sent».

Фінальна стадія обміну також відбувається паралельно, де ініціатор та власник книги отримують відправлення у своїх відділеннях, здійснюють огляд стану примірників та надсилають результат огляду на сервер. За умови відсутності претензій з обох сторін система присвоює запиту фінальний статус «Completed» та надсилає повідомлення про успішне завершення транзакції. Якщо ж зафіксовано хоча б одну скаргу, автоматично спрацьовує сценарій обробки інцидентів: система реєструє сутність exchange_complaint зі статусом «In progress» та асинхронно генерує робочий тікет для модератора платформи з метою подальшого вирішення конфлікту в ручному режимі. Спроєктована таким чином модель демонструє високу узгодженість із бізнес-логікою, враховує асинхронність дій користувачів та надійно захищає систему від потенційних збоїв на етапі логістики (рис. 2.4).

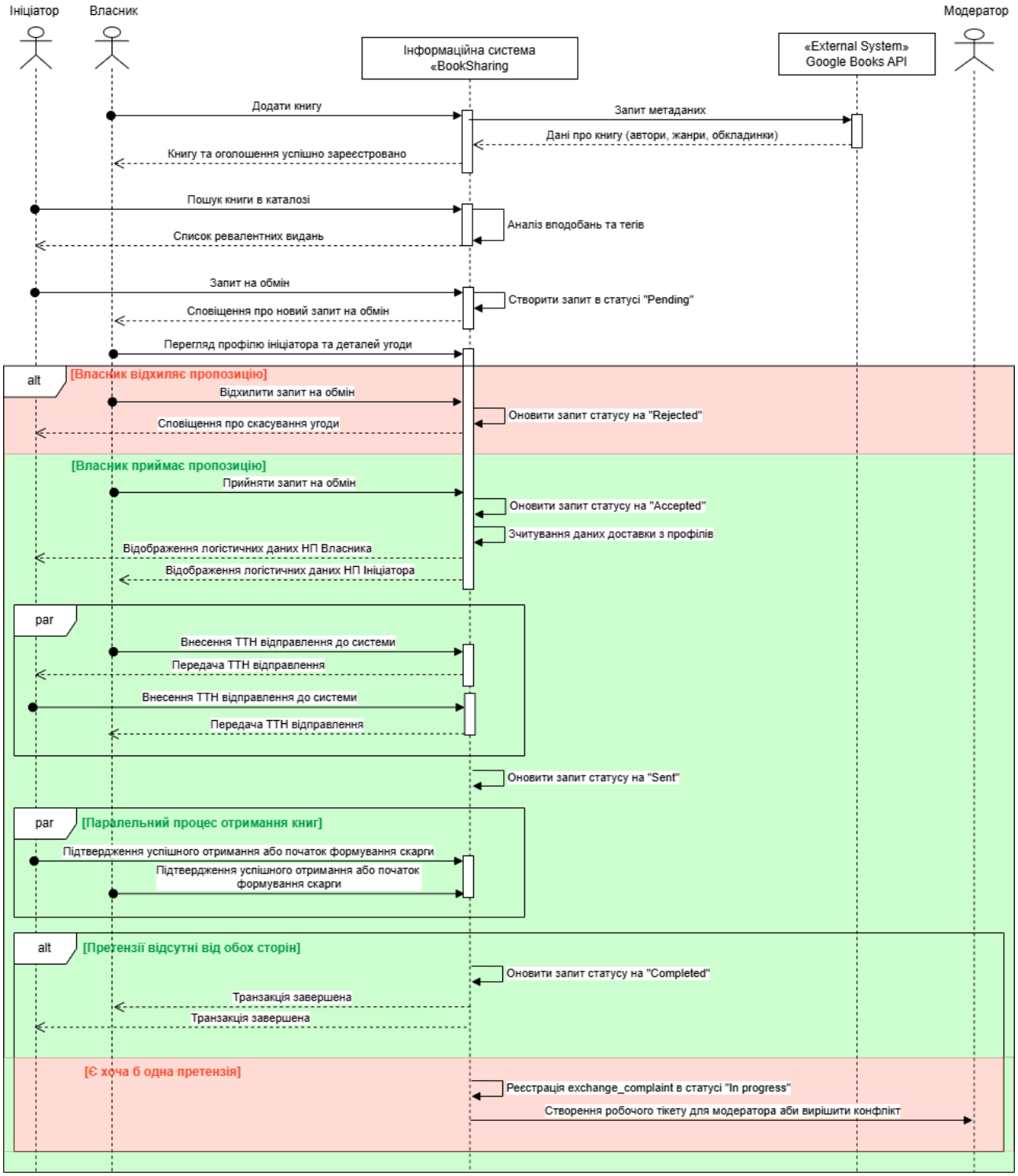

Рис. 2.4 – Діаграма послідовності інформаційної системи обліку книжкових ресурсів

2.3.4 Діаграма розгортання

Діаграма розгортання є завершальним етапом об'єктно-орієнтованого моделювання, який визначає фізичну архітектуру інформаційної системи. Вона ілюструє топологію апаратних засобів, мережеві зв'язки між ними та просторовий розподіл виконуваних програмних компонентів по відповідних обчислювальних вузлах. Для платформи «BookSharing» спроєктовано надійну трирівневу клієнт-серверну архітектуру, що забезпечує високу продуктивність, безпеку даних та модульну незалежність компонентів під час майбутнього масштабування проєкту.

Фізична інфраструктура вебзастосунка логічно декомпозирована на три основні апаратні вузли. Клієнтський рівень представлений кінцевим пристроєм користувача, на якому функціонує веббраузер як середовище виконання візуальної частини додатку. Взаємодія цього клієнтського вузла з серверною інфраструктурою відбувається через глобальну мережу Інтернет за допомогою стандартних протоколів передачі гіпертексту HTTP та захищеного HTTPS. Центральним елементом обробки бізнес-логіки виступає вебсервер, на якому розгорнуто середовище виконання мови програмування Python у зв'язці з вебфреймворком Django. Цей вузол інкапсулює в собі ключові програмні артефакти системи: файли вихідного коду проєкту з розширенням .py, систему HTML-шаблонів для динамічного рендерингу інтерфейсу, а також директорії зі статичними та медіафайлами, які включають завантажені користувачами фотографії фізичних примірників книг.

Зберігання, структуризація та забезпечення цілісності великих масивів інформації винесені на окремий ізольований вузол – сервер бази даних. У цьому обчислювальному середовищі функціонує об'єктно-реляційна система управління базами даних PostgreSQL. Вона містить у собі артефакти фізичної схеми бази даних та безпосередньо самі накопичені користувацькі і транзакційні дані. Обмін даними між вебсервером та сервером бази даних здійснюється через захищену внутрішню мережу за допомогою з'єднань TCP/IP із використанням

SQL-запитів. Такий поділ на рівень додатку та рівень даних гарантує оптимальний розподіл обчислювального навантаження, підвищує стійкість системи та дозволяє зручно налаштовувати резервне копіювання інформації (рис. 2.5).

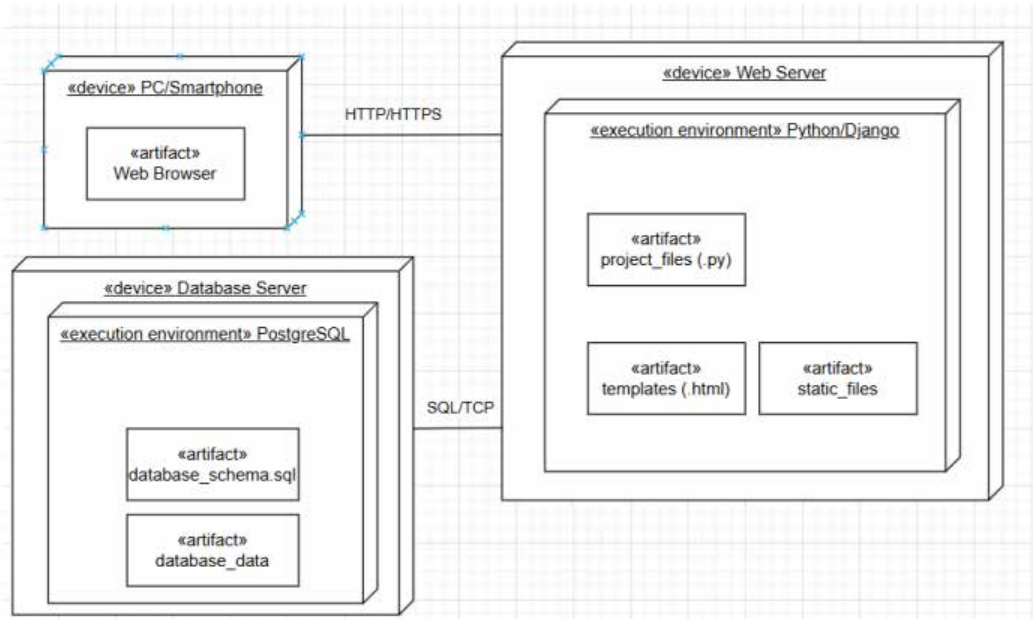

Рис. 2.5 – Діаграма розгортання інформаційної системи обліку книжкових ресурсів

2.4 Математичне забезпечення підсистеми рекомендацій

Для підвищення функціональної цінності платформи «BookSharing» та автоматизації пошуку релевантного контенту в архітектуру системи було інтегровано підсистему рекомендацій. Основне завдання цього модуля полягає в реалізації сценарію, підбору подібних книг за різними критеріями при перегляді користувачем різних пропозицій. З огляду на технічну реалізацію та структуру бази даних, базовим методом було обрано контентно-орієнтовану фільтрацію, що базується на обчисленні міри подібності між об'єктами без прив’язки до попередньої активності користувача.

Процес підготовки даних для алгоритму включає формування агрегованого текстового профілю для кожного запису в таблиці назв книг. Цей профіль створюється шляхом об'єднання назви книги, переліку пов'язаних авторів та

масиву тегів. Важливою деталлю програмної реалізації є нормалізація імен авторів. Для уникнення некоректних збігів під час розрахунків усі пробіли в іменах видаляються, що перетворює кожного автора на цілісний унікальний токен. Такий підхід дозволяє математичній моделі точніше диференціювати авторів та запобігати розмиванню результатів за загальними іменами чи прізвищами.

Для перетворення отриманих текстових профілів у векторну форму, придатну для обчислень, застосовується метод зважування TF-IDF. Цей алгоритм дозволяє виокремити найбільш значущі ознаки книги, надаючи більшої ваги рідкісним і специфічним термінам та знижуючи вагу загальновживаних слів. У випадку моєї системи – це вузькоспеціалізовані теги чи прізвища авторів. Математичне обчислення ваги W_{ij} терміна i в документі j визначається за формулою:

$$W_{ij} = TF_{i,j} \cdot IDF_i \tag{2.1}$$

де $TF_{i,j}$ відображає частоту терміна в межах одного профілю, а IDF_i обчислюється як:

$$IDF_i = \log(\frac{|D|}{|d: t_i \in d|}) \tag{2.2}$$

де $|D|$ – загальна кількість книг у базі, а знаменник відображає кількість книг, де зустрічається даний термін. Логарифмування дозволяє ефективно масштабувати значущість термінів, роблячи векторну модель стійкою до шуму.

Після трансформації всього книжкового каталогу в матрицю TF-IDF векторів, підсистема виконує пошук найбільш релевантних записів щодо цільової книги . Для цього обчислюється косинусна подібність між вектором цільового об’єкта та векторами всіх інших примірників у системі. Метрика вимірює косинус кута між векторами у багатовимірному просторі за формулою:

$$sim(A, B) = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} B_i^2}\sqrt{\sum_{i=1}^{n} A_i^2}} \tag{2.3}$$

Такий математичний апарат забезпечує точність рекомендацій та стабільну роботу системи за умов постійного розширення асортименту бібліотеки.

2.5 Аналітична візуальна панель та агрегація статистичних даних

Для забезпечення ефективного моніторингу та управління інформаційною системою «BookSharing» на етапі проєктування було передбачено створення спеціалізованої аналітичної сторінки. Головним завданням цього архітектурного модуля є безперервний збір, структуризація та візуалізація ключових показників ефективності платформи. Завдяки впровадженню підсистеми статистики модератори отримають змогу відстежувати динаміку розвитку спільноти, аналізувати активність транзакцій обміну та виявляти потенційні аномалії в поведінці користувачів. Проєктування даного компонента базується на концепціях переходу від базового транзакційного оброблення даних до елементів аналітичного оброблення, де сирі записи з реляційних таблиць трансформуються у комплексні інформаційні зведення, необхідні для прийняття управлінських рішень щодо розвитку продукту.

Теоретичною основою для формування статистичних звітів є операції реляційної алгебри, зокрема операції проєкції, вибірки та групування. Алгоритм агрегації передбачає розбиття великих масивів даних на логічні підмножини за визначеними критеріями – наприклад, за статусом запиту на обмін, дисциплінарним станом профілів, тобто кількістю страйків, або жанровою приналежністю книг. До утворених підмножин застосовуються математичні агрегатні функції. Теоретично це зводиться до обчислення кількості записів, формування частотних розподілів та аналізу часових рядів. Наприклад, для визначення найпопулярніших літературних напрямків система повинна згрупувати примірники за їхніми тегами та підрахувати частоту їхньої появи в активних запитах. Подібний алгоритмічний підхід проєктується і для аналізу логістичної активності: агрегація даних щодо обраних користувачами міст для доставки дозволить сформувати географічну статистику поширеності сервісу по регіонах.

З точки зору програмної архітектури, процес функціонування аналітичної панелі спроєктовано за принципом чіткого розмежування логіки математичних обчислень та логіки візуального представлення. Серверна частина відповідатиме за виконання складних агрегаційних SQL-запитів до бази даних. Оскільки обчислення статистики по всьому масиву даних є об’ємним процесом, архітектура передбачає оптимізацію за допомогою індексування ключових полів групування та потенційного кешування результатів, що забезпечить високу швидкодію адміністративної панелі. Сформований структурований масив статистичних даних передаватиметься на клієнтський рівень, де проєктується застосування алгоритмів візуалізації. Трансформація абстрактних числових значень у графічні об'єкти, що значно знижує когнітивне навантаження на оператора системи. Це дозволить миттєво зчитувати глобальні тренди, оцінювати відсоток успішно завершених обмінів у порівнянні з відхиленими та аналізувати криву приросту нових користувачів у часі.

2.6 Модуль модерації та система управління репутацією користувачів

Для забезпечення безпеки, мінімізації шахрайських дій та підтримки високого рівня довіри між учасниками платформи «BookSharing», на етапі проєктування було розроблено архітектуру модуля модерації та управління репутацією. Оскільки екосистема базується на моделі взаємодії «споживач-споживач» і передбачає обмін фізичними матеріальними цінностями, ймовірність виникнення конфліктних ситуацій, таких як невідповідність стану книги опису або порушення домовленостей щодо відправки, є неминучою. Для вирішення цієї проблеми було спроєктовано механізм децентралізованого контролю якості з подальшим централізованим арбітражем. Фундаментом цієї підсистеми виступає математично обґрунтована політика репутації, яка автоматизує дисциплінарний вплив на користувачів-порушників.

Теоретична модель алгоритму оскарження базується на переведенні транзакції обміну у виключний стан. Якщо ініціатор обміну під час фізичного отримання примірника виявляє його критичні дефекти або невідповідність заявленим на платформі характеристикам, система дозволяє ініціювати процес подання скарги. На рівні бази даних цей прецедент проєктується як створення нового запису у спеціалізованій транзакційній таблиці скарг, що містить ідентифікатори обох сторін угоди, текстове обґрунтування претензії та посилання на проблемну книгу. Після реєстрації скарги алгоритм тимчасово блокує закриття життєвого циклу обміну та передає управління об'єктом на рівень привілейованого актора – модератора. Модератор, виступаючи незалежним арбітром, аналізує зібрані дані та ухвалює рішення щодо відхилення скарги або її задоволення.

Ядром дисциплінарної політики платформи є детермінований алгоритм накопичення штрафних балів, або ж страйків, який захищає екосистему від систематичних зловживань. Кожен підтверджений модератором факт порушення правил призводить до інкрементування лічильника штрафів у профілі винного користувача. Математично та логічно цей механізм спроєктовано як порогову функцію активації: система постійно порівнює поточну кількість страйків користувача S із заздалегідь визначеним критичним лімітом S_{max}. Умовою безпечного функціонування профілю є виконання нерівності $S < S_{max}$. Щойно кількість штрафних балів досягає або перевищує максимально допустиме значення, спрацьовує системний тригер, який автоматично змінює логічний стан профілю. Такий архітектурний підхід дозволяє платформі саморегулюватися, автоматично ізолюючи недобросовісних контрагентів від подальшої взаємодії із загальним каталогом та іншими користувачами, тим самим підвищуючи загальну надійність програмного продукту. Хронологію взаємодії між користувачами, інтерфейсом та модулем модерації під час обробки скарг і застосування

дисциплінарних заходів наведено на діаграмі послідовності. (рис. 2.6).

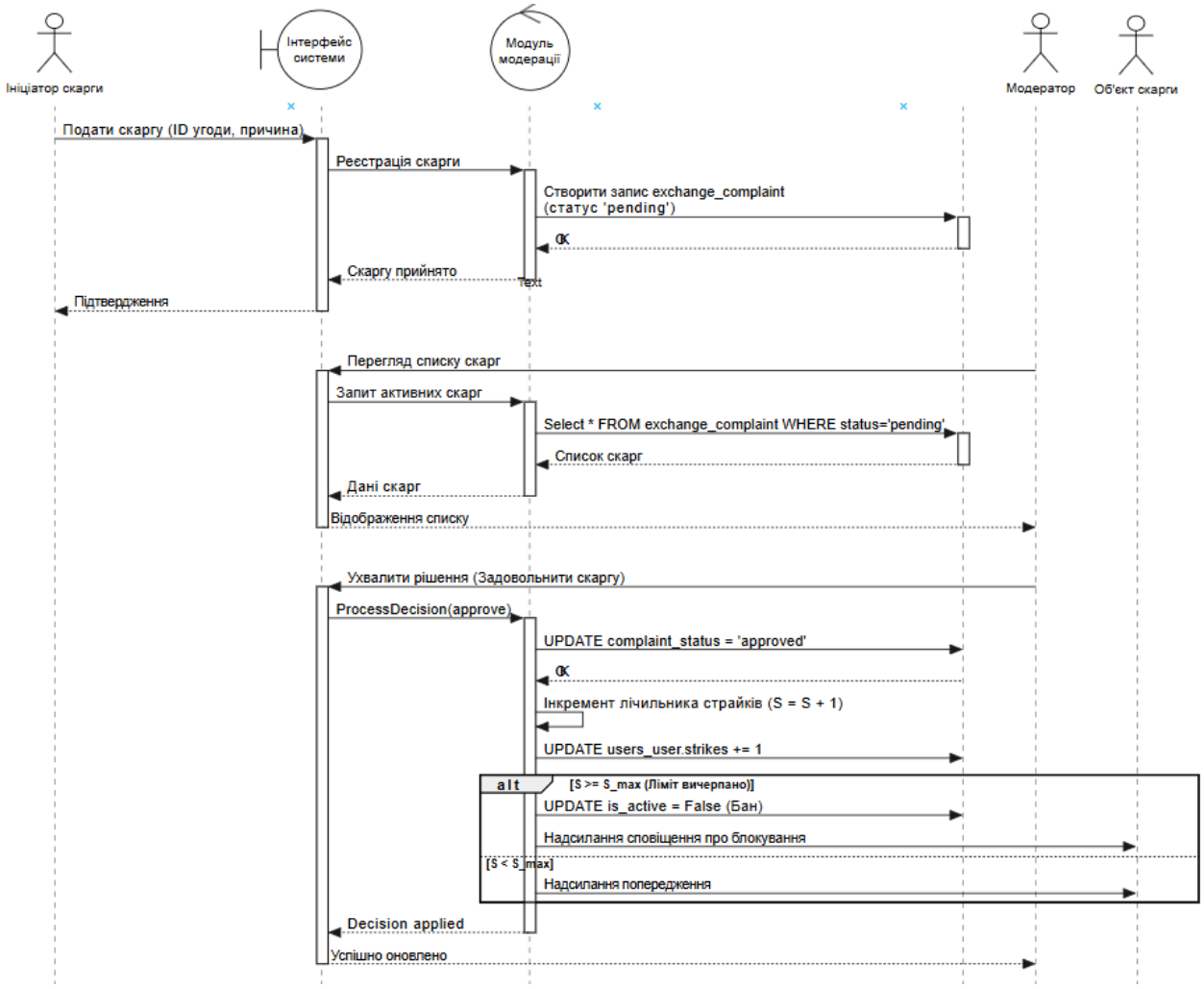

Рис. 2.6 – Діаграма послідовності процесу розгляду скарги та управління репутацією інформаційної системи обліку книжкових ресурсів

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Вибір та обґрунтування технологічного стеку розробки

Успішна реалізація багатофункціональної інформаційної системи вимагає ретельного підбору технологічного стеку, який здатен забезпечити не лише базові операції з даними, але й інтеграцію складних математичних алгоритмів. В якості основної мови програмування для розробки серверної частини було обрано високорівневу мову Python. Її вибір обґрунтовується виразним синтаксисом, високою швидкістю розробки та, що найважливіше для даного проєкту, беззаперечним лідерством в екосистемі машинного навчання та аналізу даних. Використання Python дозволило реалізувати весь життєвий цикл додатку. Це і обробка HTTP-запитів, і векторизації текстових профілів книг у єдиному мовному середовищі, без необхідності розгортання окремих мікросервісів для підсистеми рекомендацій.

Для побудови вебінфраструктури застосовано повностековий фреймворк Django, який функціонує на базі архітектурного патерну MVT. Цей інструмент надає розробнику готові надійні модулі для автентифікації користувачів, маршрутизації, захисту від поширених вебуразливостей та управління сесіями. Крім того, вбудована адміністративна панель Django стала ідеальним фундаментом для швидкого розгортання робочого місця модератора, дозволяючи зосередити ресурси на розробці унікальної бізнес-логіки. Взаємодія з базою даних у межах фреймворку здійснюється через механізм об'єктно-реляційного відображення Django ORM, який абстрагує прямі SQL-запити у методи об'єктів Python, підвищуючи безпеку та полегшуючи подальшу підтримку коду.

Основним сховищем даних виступає об'єктно-реляційна система управління базами даних PostgreSQL. Її інтеграція в проєкт зумовлена високою надійністю, повною відповідністю принципам ACID та здатністю ефективно обробляти

складні агрегаційні запити. У зв'язці з PostgreSQL система забезпечує строгу цілісність даних на рівні зовнішніх ключів, що гарантує безпомилкове зв'язування профілів користувачів, книг, скарг та логістичних записів.

Реалізація підсистеми рекомендацій виконана за допомогою відкритої бібліотеки машинного навчання «scikit-learn». Вона надає інструменти для роботи з багатовимірними матрицями та виконання складних математичних обчислень. Клієнтська частина платформи реалізована з використанням класичної тріади вебтехнологій: HTML5 для семантичної розмітки, CSS3 для візуального стилізування та JavaScript для забезпечення динамічної взаємодії в браузері. Для пришвидшення розробки адаптивного користувацького інтерфейсу, здатного коректно відображатися як на десктопних, так і на мобільних пристроях, було використано CSS-фреймворк, який надав готову сітку та компоненти для формування зручного каталогу книг і форм обміну.

3.2 Розробка серверної частини та структури бази даних

Серверна частина системи «BookSharing» побудована за модульним принципом, що реалізовано через розподіл функціоналу між декількома незалежними додатками Django: users, books та exchange. Така архітектура дозволяє досягти високого рівня інкапсуляції коду та полегшує подальшу підтримку системи. Кожен додаток має власну модельну схему, що відображає певну частину бізнес-логіки проекту. Взаємодія між модулями забезпечується за допомогою механізмів зовнішніх ключів та зв'язків «багато-до-багатьох».

Центральне місце в системі займає додаток, що відповідає за управління користувачами. У ньому реалізовано кастомну модель CustomUser (рис. 3.1), яка успадковується від базового класу AbstractUser. Ця модель розширює стандартні можливості фреймворку, додаючи специфічні атрибути: прапорець модератора, лічильник штрафних балів для системи управління репутацією, а також географічну прив'язку, що є критично важливим для інтеграції з логістичними сервісами.

```python
class CustomUser(AbstractUser):  5 usages  ± beekugan
    is_moderator = models.BooleanField(default=False, verbose_name="Модератор")
    interested_tags = models.ManyToManyField(
        to 'books.Tag',
        blank=True,
        related_name='interested_users',
        verbose_name="Цікаві теми"
    )
    phone_number = models.CharField(max_length=15, blank=True, null=True, verbose_name="Телефон")
    avatar = models.ImageField(upload_to='avatars/', blank=True, null=True, verbose_name="Аватар")
    middle_name = models.CharField(max_length=50, blank=True, null=True, verbose_name="По батькові")
    city = models.CharField(max_length=100, blank=True, null=True, verbose_name="Місто")
    nova_poshta_branch = models.CharField(max_length=255, blank=True, null=True, verbose_name="Відділення Нової Пошти")
    strikes = models.IntegerField(default=0, verbose_name="Штрафні бали")

    def __str__(self):  ± beekugan
        return self.username
```

Рис. 3.1 – Програмна реалізація розширеної моделі користувача

Додаток books відповідає за управління каталогом літератури та містить найбільш розгалужену реляційну структуру в системі. Ключовим архітектурним патерном тут є чітке розділення між абстрактним поняттям видання та його фізичним втіленням. Модель Title (рис.3.2) акумулюватиме нормалізовані метадані, зв'язані через відношення «багато-до-багатьох» із допоміжними довідниками: Author, Genre та Tag.

```python
class Title(models.Model):  19 usages  ± beekugan

    isbn = models.CharField(max_length=13, unique=True, verbose_name="ISBN", null=True, blank=True)
    name = models.CharField(max_length=255, verbose_name="Назва книги")
    publication_year = models.PositiveIntegerField(verbose_name="Рік видання", null=True, blank=True)

    publisher = models.ForeignKey(Publisher, on_delete=models.SET_NULL, null=True, blank=True,
                                  verbose_name="Видавництво")

    authors = models.ManyToManyField(Author, related_name='titles', verbose_name="Автори")
    genres = models.ManyToManyField(Genre, related_name='titles', verbose_name="Жанри")

    cover_image = models.ImageField(upload_to='catalog_covers/', verbose_name="Обкладинка", blank=True, null=True)
    tags = models.ManyToManyField(Tag, related_name='titles', blank=True, verbose_name="Теги (для AI рекомендацій)")
    description = models.TextField(blank=True, null=True)
    def __str__(self):  ± beekugan
        return self.name

    class Meta:  ± beekugan
        verbose_name = "Книга (Каталог)"
        verbose_name_plural = "Книги (Каталог)"

    is_reviewed = models.BooleanField(
        default=False,
        verbose_name="Перевірено модератором",
        help_text="Зніміть галочку, якщо книга додана автоматично і потребує дозаповнення"
    )
```

Рис. 3.2 – Фрагмент програмної реалізації каталогу книг (Title)

Фізичний примірник, що належить конкретному користувачеві, описується моделлю Copy (рис. 3.3). Вона містить зовнішні ключі на власника та абстрактну книгу, а також інкапсулює бізнес-логіку самої транзакції. Завдяки атрибуту offer_type система підтримує кілька сценаріїв взаємодії: класичний обмін, безоплатне дарування або передачу за благодійний донат.

```
class Copy(models.Model):  22 usages  ± beekugan *

    CONDITION_CHOICES = [
        ('new', 'Нова'),
        ('good', 'Хороша'),
        ('fair', 'Задовільна'),
        ('bad', 'Погана'),
    ]
    STATUS_CHOICES = [
        ('available', 'Доступна'),
        ('transferred', 'Передана/Обміняна'),
    ]
    OFFER_CHOICES = [
        ('exchange', 'Обмін'),
        ('gift', 'Дарування'),
        ('donate', 'За донат'),
    ]
    title = models.ForeignKey(Title, on_delete=models.CASCADE, related_name='copies', verbose_name="Книга з каталогу")
    owner = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='copies',
                              verbose_name="Власник")
    offer_type = models.CharField(
        max_length=20,
        choices=OFFER_CHOICES,
        default='exchange',
        verbose_name="Тип пропозиції"
    )
    condition_description = models.TextField(verbose_name="Опис стану", blank=True)
    condition = models.CharField(max_length=20, choices=CONDITION_CHOICES, default='good', verbose_name="Стан книги")
    photo = models.ImageField(upload_to='copies_photos/', verbose_name="Живе фото", blank=True, null=True)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='available', verbose_name="Статус")
```

Рис. 3.3 – Фрагмент програмної реалізації каталогу книг (Copy)

3.3 Програмна імплементація підсистеми машинного навчання та рекомендацій

Важливою частиною аналітичних можливостей системи «BookSharing» є підсистема рекомендацій, яка забезпечує користувачів релевантними пропозиціями літератури на основі контентно-орієнтованої фільтрації. Програмна імплементація цієї підсистеми реалізована у вигляді окремого сервісного модуля recommender.py, який інкапсулює логіку підготовки даних, векторизації та обчислення математичної подібності. Такий архітектурний підхід

дозволяє ізолювати складні обчислення від контролерів фреймворку Django, забезпечуючи чистоту коду та можливість його повторного використання.

Підхід, використаний у розробці модуля, базується на класичних методах обробки природної мови та інформаційного пошуку, які детально описані у фундаментальних працях з машинного навчання, зокрема в роботах Крістофера Меннінга щодо векторного простору моделей та TF-IDF зважування [17, с. 31]..

Нижче наведено програмний код функції генерації рекомендацій (рис. 3.4).

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from .models import Title


def get_recommendations(target_book_id, top_n=4):  4 usages  & beekugan

    books = list(Title.objects.all().prefetch_related( *lookups: 'authors', 'tags'))

    if not books or len(books) < 2:
        return []

    book_ids = []
    combined_features = []

    for book in books:
        book_ids.append(book.id)

        authors = " ".join([author.name.replace(" ", "") for author in book.authors.all()])

        tags = " ".join([tag.name for tag in book.tags.all()])

        features = f"{book.name} {authors} {tags}"
        combined_features.append(features)

    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(combined_features)

    cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

    try:
        target_idx = book_ids.index(target_book_id)
    except ValueError:
        return []

    sim_scores = list(enumerate(cosine_sim[target_idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    top_indices = [i[0] for i in sim_scores[1:top_n + 1]]

    recommended_ids = [book_ids[i] for i in top_indices]

    recommendations = []
    for rid in recommended_ids:
        book_obj = next((b for b in books if b.id == rid), None)
        if book_obj:
            recommendations.append(book_obj)

    return recommendations
```

Рис. 3.4 – Програмна реалізація алгоритму рекомендацій

Програмна імплементація алгоритму включає кілька критичних оптимізацій. На етапі вилучення даних із СУБД використовується метод prefetch_related. Це дозволяє системі завантажити всі пов'язані теги та авторів за допомогою мінімальної кількості SQL-запитів, ефективно вирішуючи класичну проблему «N+1 запитів», що виникає при ітерації по зв'язках «багато-до-багатьох».

На етапі попередньої обробки даних алгоритм динамічно формує агрегований текстовий профіль кожної книги.

Після перетворення текстового масиву у розріджену матрицю ознак та обчислення косинусної подібності, функція виконує сортування індексів. Алгоритм зрізу гарантує, що цільова книга буде виключена з підсумкової вибірки. Фінальний блок коду виконує зворотне відображення знайдених ідентифікаторів на реальні об'єкти моделі Title, формуючи готовий список для передачі на клієнтський інтерфейс.

3.4 Розробка клієнтського інтерфейсу та аналітичної панелі

Клієнтська частина платформи «BookSharing» розроблена з фокусом на інтуїтивну зрозумілість, адаптивність та забезпечення безшовного користувацького досвіду.

Значна увага під час розробки була приділена підсистемі безпеки та ідентифікації. Окрім класичної локальної системи автентифікації, в архітектуру інтегровано протокол OAuth 2.0 для забезпечення входу через облікові записи Google (рис.3.5). Такий гібридний підхід значно знижує бар'єр входу для нових користувачів. Для захисту екосистеми від автоматизованого спаму та створення фейкових акаунтів, у локальній системі реєстрації реалізовано механізм обов'язкового підтвердження електронної пошти за допомогою токенів, що генеруються сервером і відправляються користувачеві листом. Також реалізована функція скидання пароля.

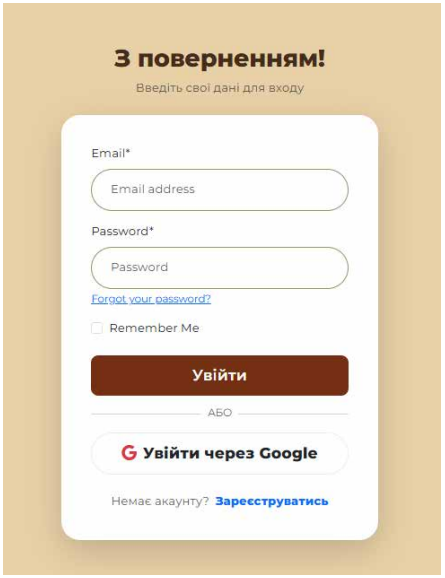

Рис. 3.5 – Форма авторизації користувача до інформаційної системи обліку книжкових ресурсів

Після успішної авторизації користувач отримує доступ до особистого кабінету. Інтерфейс профілю спроєктовано як єдиний центр управління діяльністю на платформі. Тут відображені персональні дані та надається швидкий доступ до управління власними примірниками книг і активними запитами на обмін.

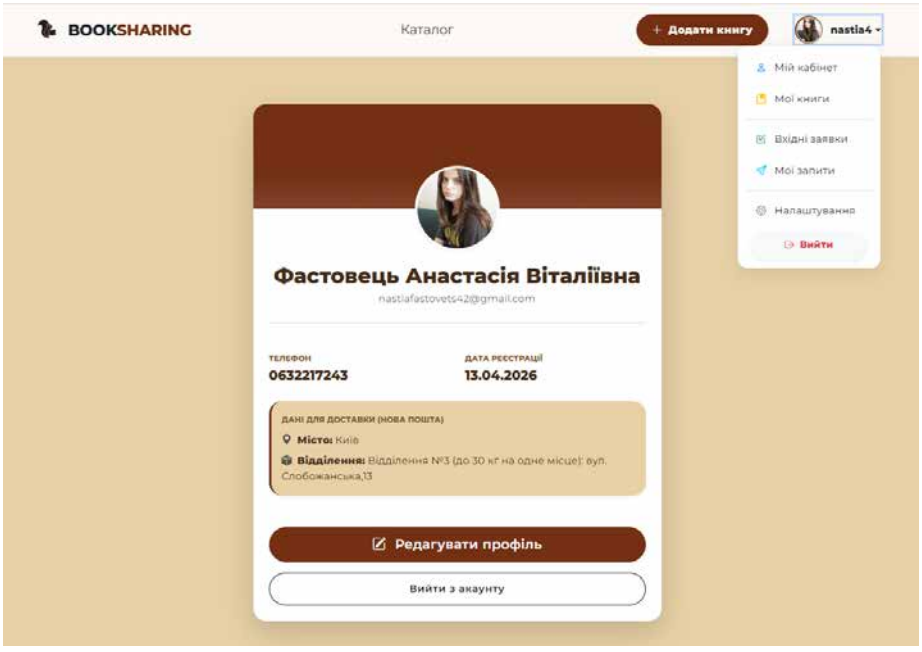

Рис. 3.6 – Особистий кабінет користувача системи, а також зручна навігація по особистих даних

Основним бізнес-процесом системи є пошук літератури та ініціація угоди обміну. Цей потік починається на сторінці глобального каталогу (рис.3.7), де реалізовано механізми поділу великих обсягів контенту та багатофакторної фільтрації за авторами, жанрами і тегами.

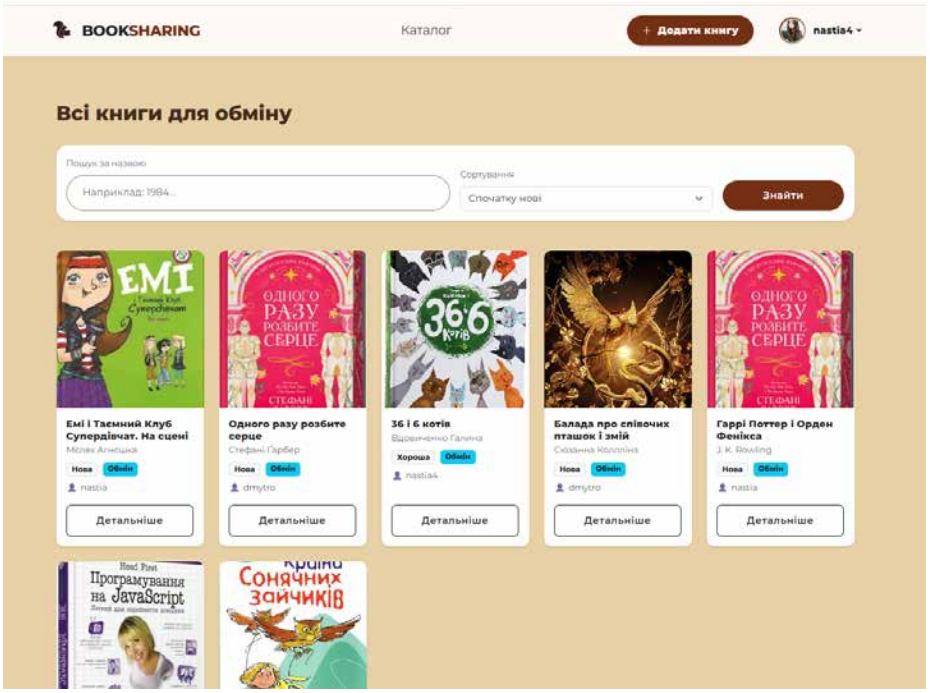

Рис 3.7 – Глобальний каталог літератури з панеллю фільтрації

Перейшовши на сторінку конкретної книги, користувач бачить її детальний опис, інформацію про доступні фізичні примірники від інших учасників та блок персоналізованих рекомендацій (рис. 3.8), згенерованих підсистемою машинного навчання.

Рис. 3.8 – Детальна сторінка оголошення та блок AI-рекомендацій

Для ініціації транзакції користувач натискає відповідну кнопку біля обраного примірника, після чого система генерує інтерактивну форму запиту на обмін. У цій формі контрагенти можуть узгодити деталі логістики та підтвердити готовність до відправлення.

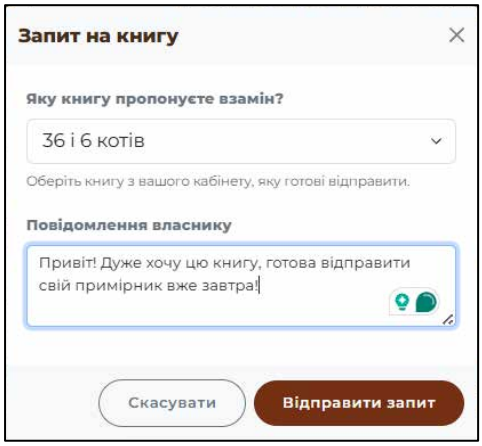

Рис. 3.9 – Інтерфейс оформлення запиту на обмін літературою

Після ініціації угоди подальше управління її життєвим циклом здійснюється через спеціалізований розділ управління транзакціями. Для забезпечення чіткого розмежування зон відповідальності цей інтерфейс логічно поділено на дві категорії: «Мої запити» та «Вхідні запити». У цьому розділі відображається

перелік усіх активних та історичних транзакцій із візуальною індикацією їхнього поточного системного статусу.

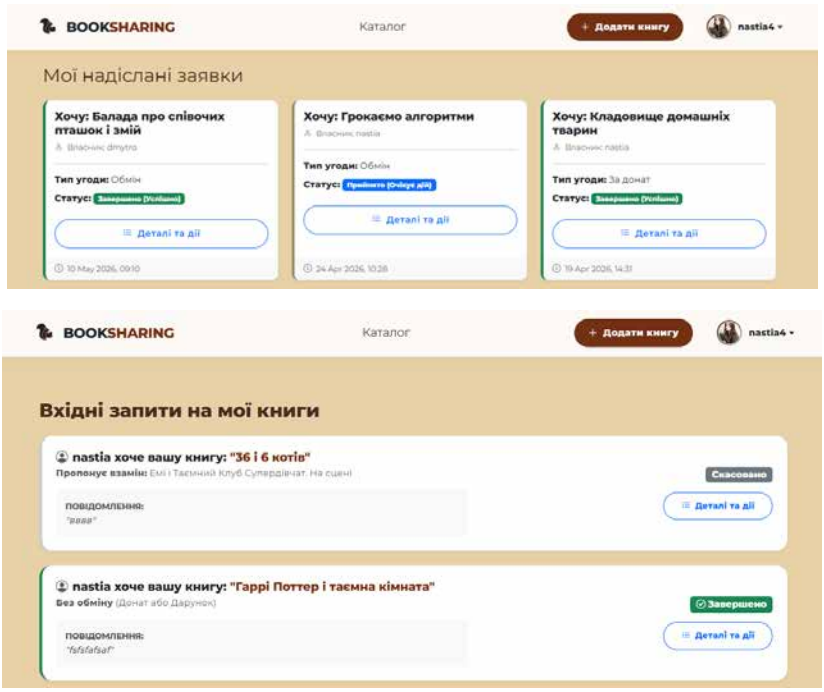

Рис. 3.10 – Інтерфейс керування вхідними та вихідними заявками користувача

Кожна транзакція має окрему сторінку деталей угоди (рис. 3.11), яка виступає спільним інтерактивним робочим простором для обох контрагентів. Бізнес-процес обміну програмно спроєктовано як послідовний кінцевий автомат. На першому етапі власник книги розглядає вхідний запит і має змогу його підтвердити або відхилити. У разі підтвердження угоди система переводить транзакцію на етап логістичної обробки. Користувач-відправник, після фізичного оформлення відправлення, зобов'язаний внести у систему номер ТТН. Ця дія змінює статус примірника на «В дорозі» та відкриває для користувача-отримувача фінальну дію – можливість підтвердити успішне фізичне отримання книги. Лише після програмного підтвердження отримання від обох сторін життєвий цикл транзакції вважається успішно закритою, а об'єкт книги остаточно змінює свого власника у реляційній базі даних.

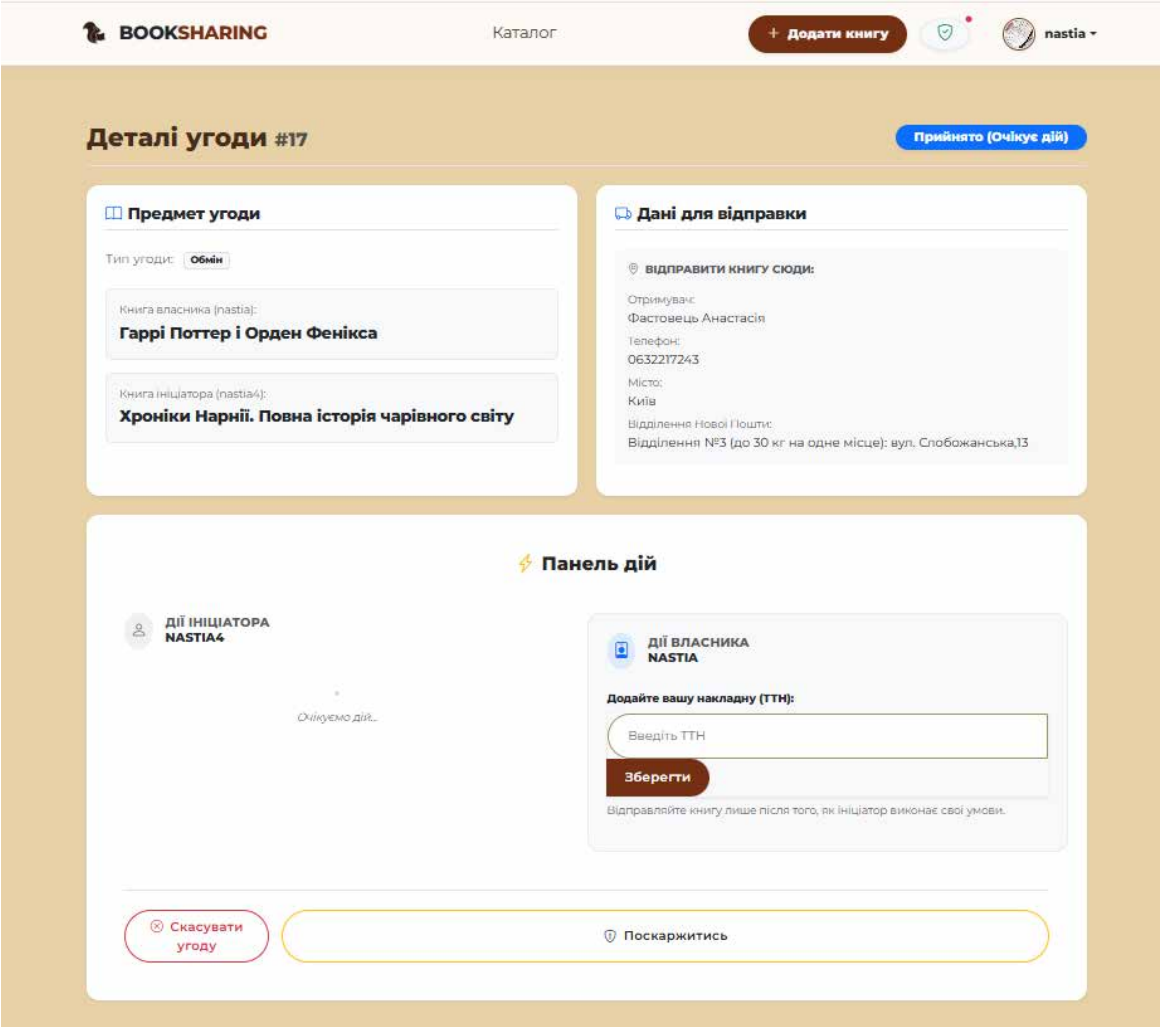

Рис. 3.11 – Сторінка деталей транзакції та логістичного супроводу обміну

Окрема увага приділена розробці інструментів адміністративного контролю. У системі реалізовано дворівневу архітектуру управління: глобальне адміністрування та оперативна модерація. Глобальне адміністрування здійснюється через стандартну панель Django Admin (рис.3.12), яка надає користувачам з доступом повний CRUD-доступ до всіх таблиць бази даних.

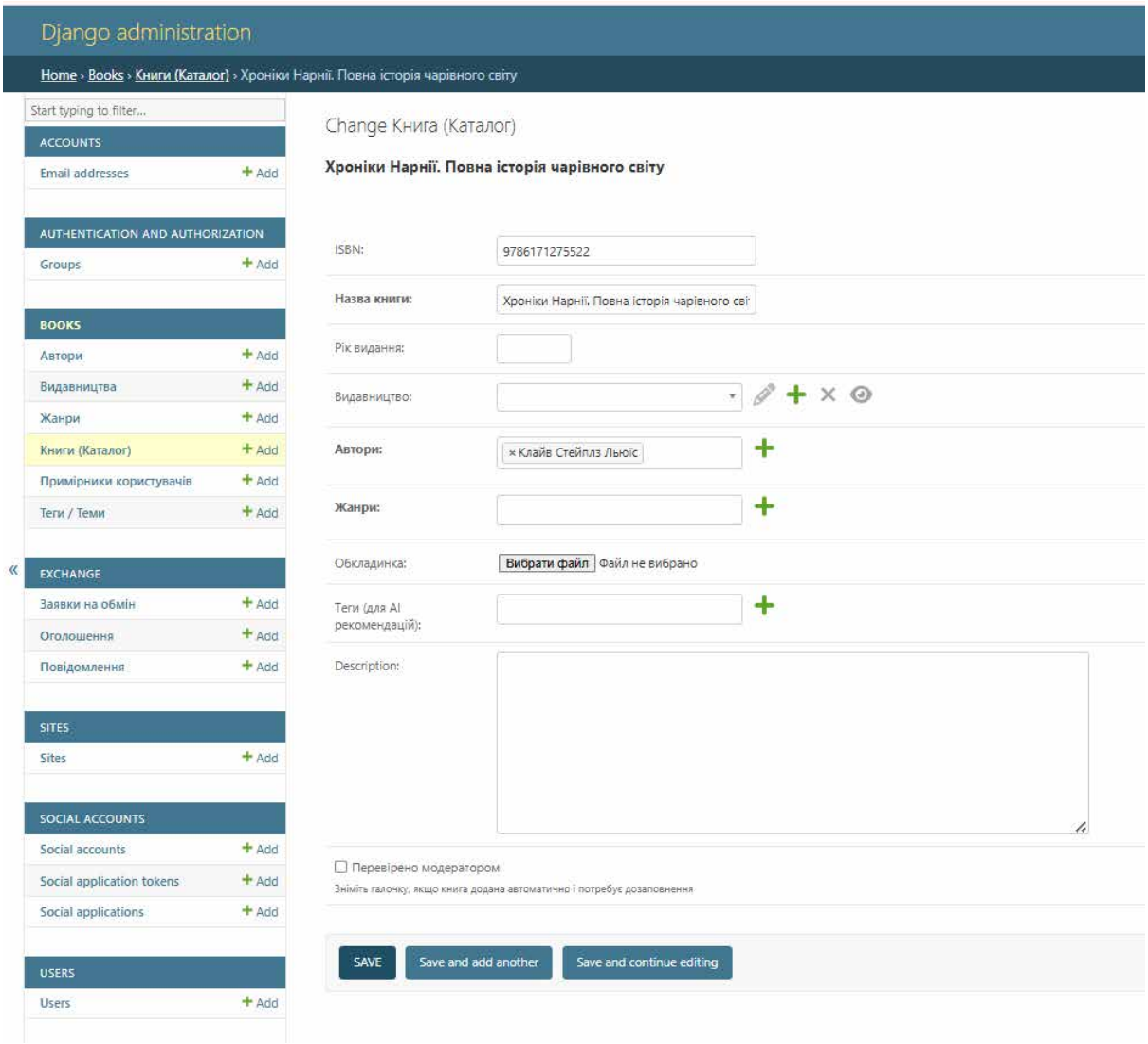

Рис. 3.12 – Панель глобального адміністрування

Для оперативного вирішення щоденних завдань було розроблено спеціалізований кабінет модератора. Основними функціями кабінету модератора є черга нових книг, які потребують затвердження від модератора. А також панель розгляду скарг. У панелі скарг модератор (рис. 3.13) може ознайомитися з претензіями сторін та в один клік ухвалити рішення про нарахування штрафного бала порушнику.

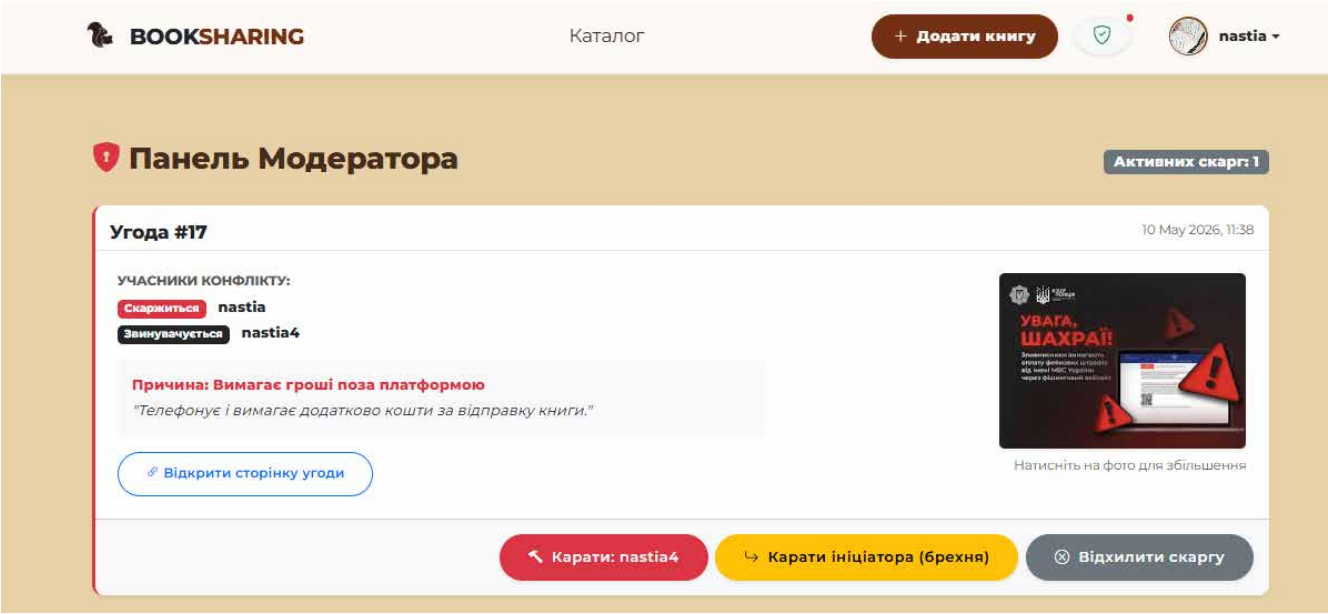

Рис. 3.13 – Спеціалізований інтерфейс модератора платформи

Для моніторингу загального стану екосистеми в інтерфейс інтегровано аналітичну панель (рис. 3.14). Вона генерує ключові статистичні показники та візуалізує їх у вигляді інтерактивних графіків, що дозволяє адміністрації приймати обґрунтовані рішення щодо розвитку проєкту.

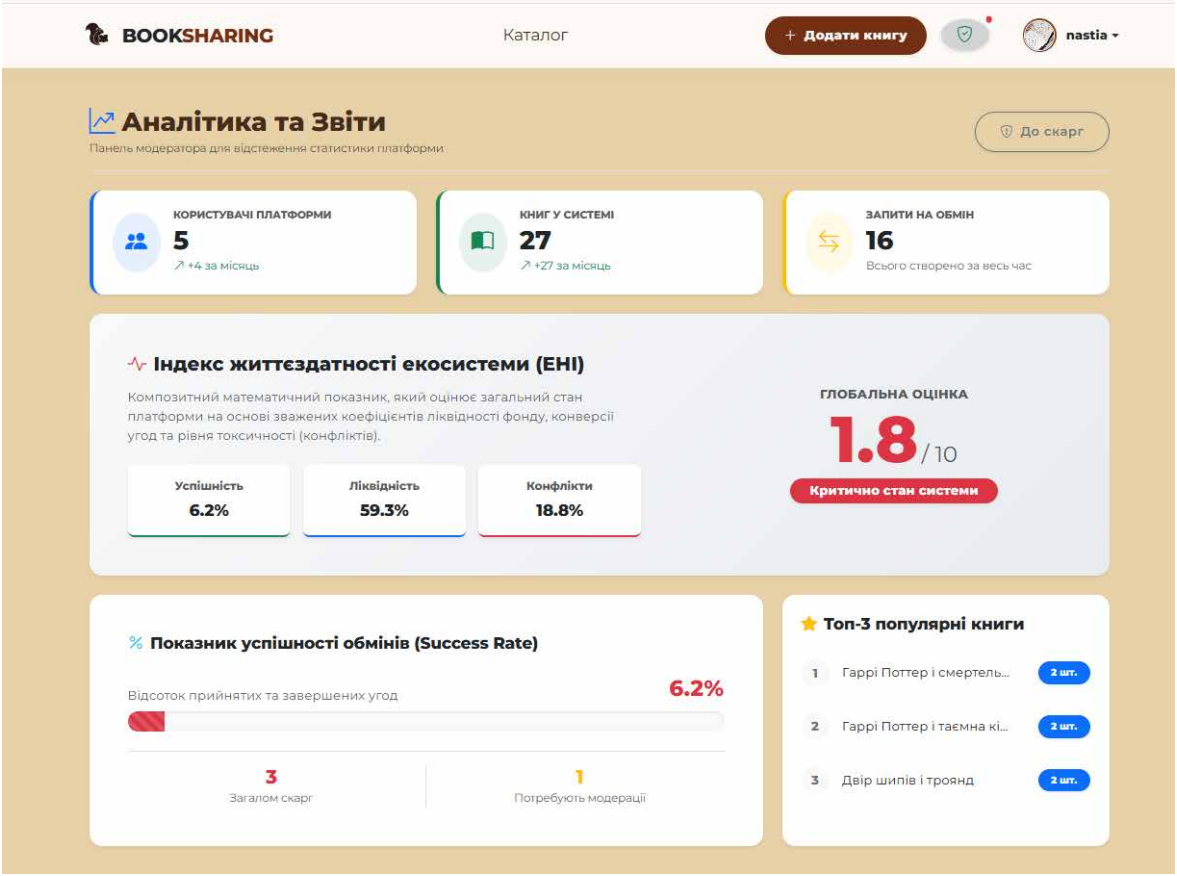

Рис. 3.14 – Аналітична панель показників ефективності платформи

3.5 Програмна інтеграція зовнішніх сервісів

Сучасні вебзастосунки проєктуються з урахуванням парадигми відкритої архітектури, що передбачає активне використання API для розширення власного функціонала. У межах розробки серверної частини платформи «BookSharing» було реалізовано інтеграцію двох ключових зовнішніх інфраструктурних рішень: бібліографічного сервісу Google Books та логістичного шлюзу компанії «Нова Пошта». Ці інтеграції автоматизують рутинні процеси, підвищують узгодженість даних та значно покращують загальний користувацький досвід.

Першим напрямком інтеграції є взаємодія з глобальним сервісом Google Books API. Програмне рішення щодо залучення цього сервісу зумовлене потребою у стандартизації каталогу літератури та мінімізації помилок під час ручного введення даних про книги. Розробка архітектурного рішення для цього модуля, а також проєктування форматів асинхронних клієнт-серверних RESTful-запитів до публічного шлюзу виконувалися відповідно до специфікацій та технічних протоколів, представлених в офіційній документації сервісу [16]. Коли користувач на сторінці додавання нової книги починає вводити її назву, клієнтський скрипт перехоплює введені символи та відправляє запит на бекенд платформи.

Отриманий від зовнішнього сервера масив даних у форматі JSON парситься, тобто система вилучає перевірені метадані та повертає їх до клієнтської частини. Користувач бачить ці результати у вигляді зручного випадаючого списку (рис. 3.15). Якщо обраної книги ще немає в локальній базі даних платформи, система автоматично створює новий запис у таблиці каталогу, але присвоює йому статус неперевіреного, що передає об'єкт у чергу на модерацію. Це підвищує якість контенту та запобігає засміченню бази даних дублікатами.

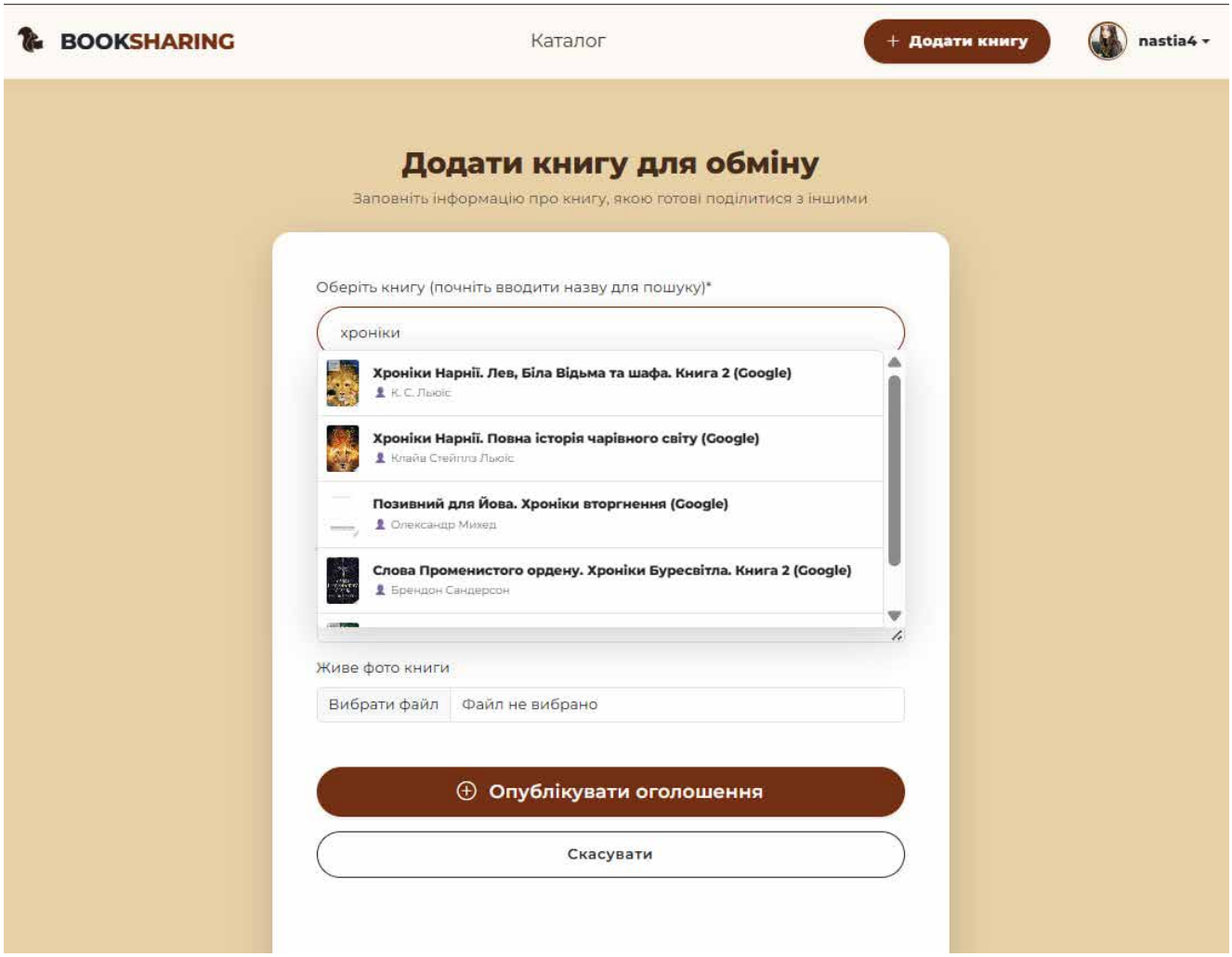

Рис. 3.15 – Інтерфейс додавання книги з використанням асинхронного пошуку через Google Books API

Другим критично важливим інфраструктурним компонентом є логістичний модуль, побудований на базі інтеграції з API поштового оператора «Нова Пошта». Оскільки фінальною метою кожної угоди на платформі є фізичний обмін примірниками, забезпечення користувачів актуальною інформацією щодо точок доставки є пріоритетним завданням. Програмна взаємодія з логістичним шлюзом та архітектурне проєктування форматів передачі даних здійснюються на основі захищених HTTP-запитів із використанням унікального криптографічного ключа доступу розробника, згідно з технічним регламентом та інструкціями офіційного порталу розробника компанії [14].

Для оптимізації швидкодії та уникнення лімітів на кількість зовнішніх мережевих викликів, система періодично синхронізує довідники населених

пунктів і діючих відділень, зберігаючи їх у локальній базі даних. Завдяки цій інтеграції, на етапі налаштування профілю або підтвердження угоди обміну, користувачі одразу отримують актуальні дані. Наприклад, при виборі міста, скрипт автоматично підтягує відповідний масив доступних відділень із бази даних і формує випадаючий список (рис.3.16). Це повністю виключає ймовірність друкарських помилок при вказуванні адреси доставки та пришвидшує процес оформлення логістичних накладних.

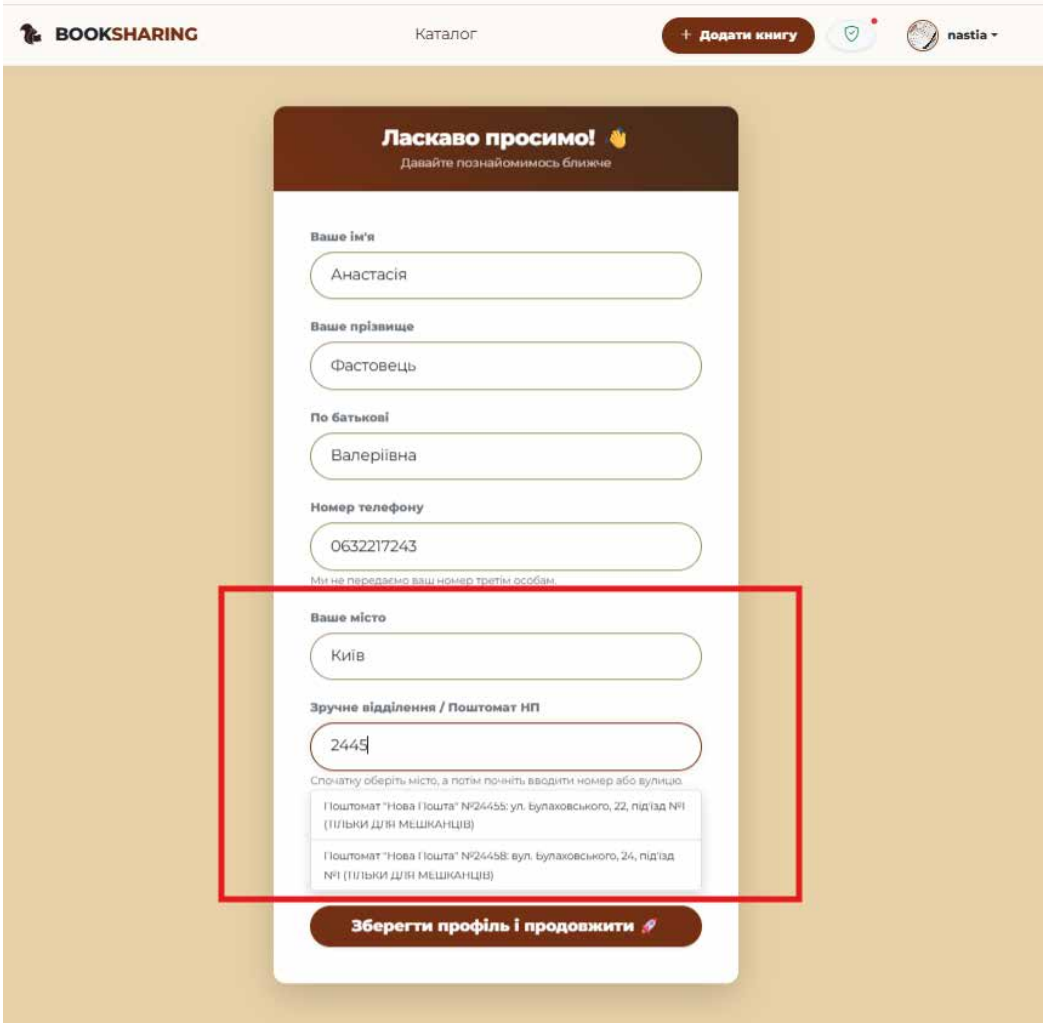

Рис. 3.16 – Інтерфейс вибору відділення доставки за допомогою інтеграції з API «Нова Пошта»

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

4.1 Методика та середовище тестування

Для забезпечення високої якості та надійності програмної платформи «BookSharing» було розроблено та застосовано комплексну методику тестування. Враховуючи багатокомпонентну архітектуру системи, яка включає підсистему машинного навчання, реляційну базу даних та зовнішні API-інтеграції, процес верифікації було розділено на кілька незалежних етапів: модульне, інтеграційне та функціональне тестування.

Модульне тестування застосовувалося для перевірки ізольованих компонентів серверної частини, зокрема бізнес-логіки моделей даних та математичного апарату рекомендаційної системи. Для реалізації цього етапу використовувався вбудований інструментарій фреймворку Django – клас django.test.TestCase, який базується на стандартній бібліотеці «Python unittest». Цей підхід дозволив створювати ізольовані тестові бази даних, що гарантує відсутність впливу тестових скриптів на реальні дані користувачів.

Інтеграційне та функціональне тестування проводилося за методом «Black-box testing» з метою імітації реальної поведінки користувачів та модераторів. Основний фокус на цьому етапі було зосереджено на перевірці коректності взаємодії платформи із зовнішніми шлюзами, а також на дотримання вимог користувацьких сесій і механізмів авторизації.

Середовище тестування було розгорнуто на базі локальної конфігурації, що включала:

- Операційна система Windows.
- Серверна частина, а саме будований WSGI-сервер Django, інтерпретатор мови Python версії 3.x.
- Тимчасову базу даних для тестів SQLite.

- Вебоглядач Google Chrome з використанням інструментів розробника для моніторингу мережевих запитів та інспектування DOM-дерева.

Такий комплексний підхід до організації середовища та вибору методик дозволив покрити тестами всі критичні вузли платформи та забезпечити стабільну роботу системи перед етапом її дослідного впровадження.

4.2 Модульне тестування

Модульне тестування виступає першим і базовим рівнем контролю якості платформи «BookSharing». Головна роль цього етапу полягає в ізольованій перевірці критично важливої серверної бізнес-логіки та алгоритмів без необхідності розгортання клієнтського інтерфейсу. Такий підхід гарантує, що фундаментальні процеси системи працюють безпомилково на рівні бази даних.

Для реалізації тестування використовувався вбудований інструментарій «django.test.TestCase». Під час запуску скриптів система автоматично генерувала тимчасову базу даних SQLite, що забезпечило високу швидкість виконання операцій та унеможливило пошкодження реальних даних користувачів. Конфігурування середовища тестування, автоматизація збору написаних тест-кейсів, а також візуалізація фінальних консольних звітів виконувалися із залученням інструментарію розширення можливостей тестування на базі офіційних специфікацій та технічних протоколів фреймворку Pytest [13].

Процес модульного тестування було сфокусовано на двох найважливіших модулях системи. Першим напрямком стала перевірка підсистеми управління користувачами та модерації. Тестові сценарії підтвердили коректність ініціалізації профілів та надійність механізму інкрементації страйків при фіксації порушень (рис. 4.1).

```python
from django.test import TestCase
from django.contrib.auth import get_user_model

User = get_user_model()

class CustomUserModelTest(TestCase):  new *
    def setUp(self):  new *
        self.user = User.objects.create_user(
            username='testuser',
            password='testpassword123',
            city='Київ'
        )
    def test_user_creation(self):  new *
        self.assertEqual(self.user.username,  second: 'testuser')
        self.assertEqual(self.user.city,  second: 'Київ')
        self.assertTrue(self.user.is_active)

    def test_default_strikes_is_zero(self):  new *
        self.assertEqual(self.user.strikes,  second: 0)

    def test_add_strike(self):  new *
        self.user.strikes += 1
        self.user.save()
        self.assertEqual(self.user.strikes,  second: 1)
```

Рис 4.1 – Модульне тестування підсистеми модерації користувачів

Другим, найбільш технічно складним напрямком тестування, стала верифікація алгоритму семантичних рекомендацій. Оскільки якість рекомендацій безпосередньо впливає на користувацький досвід, було розроблено сценарій, що імітує вибір літератури.

У тестовій базі було створено три об'єкти: дві споріднені книги в однаковому жанрі, зі спільним автором та тегами. Паралельно був створений кардинально інший екземпляр. Тест успішно підтвердив, що математичний апарат працює з високою точністю: алгоритм ідентифікував релевантну книгу та повністю відкинув невідповідну (рис. 4.2).

```python
def test_get_recommendations(self):  new *
    recs = get_recommendations(self.book1.id, top_n=1)
    self.assertTrue(len(recs) > 0)
    self.assertEqual(recs[0].id, self.book2.id)
    self.assertNotIn(self.book3, recs)
```

Рис. 4.2 – Тестування точності алгоритму AI-рекомендацій

Виконання пакетів тестів у консолі розробника продемонструвало відсутність логічних помилок. Усі сценарії завершилися з позитивним статусом (рис. 4.3).

```
(.venv) PS C:\Users\nasti\PycharmProjects\PythonProject\BookSharing> python manage.py test users
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 3.028s

OK
Destroying test database for alias 'default'...
```

Рис. 4.3 Результат виконання модульних тестів однієї з підсистем

4.3 Функціональне тестування користувацьких сценаріїв

Після успішної валідації внутрішньої бізнес-логіки на рівні модульних тестів, тестування перейшло на етап перевірки користувацьких Цей етап є критично важливим, оскільки він дозволяє оцінити коректність роботи системи в цілому, враховуючи взаємодію між клієнтською та серверною частинами, базою даних та зовнішніми API.

Функціональне тестування виконувалося вручну у середовищі сучасних веббраузерів. Головною метою було імітувати реальний шлях від моменту первинної реєстрації на платформі до успішного завершення транзакції обміну книгами.

Особлива увага під час тестування приділялася перевірці інтеграційних точок, а також коректності відображення статусів складних багатоетапних бізнес-процесів. Результати перевірки ключових сценаріїв задокументовані у вигляді таблиці тест-кейсів (таблиця 4.1).

Таблиця 4.1

Тест-кейси функціонального тестування системи

№	Сценарій	Дії користувача	Очікуваний результат	Результат
1	Авторизація OAuth 2.0	Натискання кнопки «Увійти через Google», вибір облікового запису.	Успішна авторизація, автоматичне створення профілю в БД, переадресація в особистий кабінет.	Відповідає очікуваному.
2	Інтеграція Google Books API	Введення назви неіснуючої в БД книги у форму додавання каталогу.	Асинхронний виклик API, відображення випадаючого списку з релевантними обкладинками та авторами.	Відповідає очікуваному.
3	Створення примірника	Заповнення форми додавання примірника, вибір стану книги, вибір типу пропозиції.	Створення об'єкта Copy, присвоєння йому дефолтного статусу available, відображення в загальному каталозі.	Відповідає очікуваному.
4	Логістична інтеграція (Нова Пошта)	Перехід до налаштувань профілю, вибір міста «Київ».	Динамічне завантаження списку активних відділень обраного міста через API логістичного оператора.	Відповідає очікуваному.
5	Життєвий цикл угоди	Ініціація запиту отримувачем, після підтвердження власником та введення ТТН. Після підтвердження отримання.	Послідовна та коректна зміна статусів транзакції на кожному етапі. Фінальний статус угоди «Завершено».	Відповідає очікуваному.
6	Модерація контенту	Модератор переходить у панель скарг та натискає кнопку «Видати страйк».	Інкрементація поля strikes у користувача-порушника, зміна статусу скарги на оброблену.	Відповідає очікуваному.

Проведене функціональне тестування підтвердило цілісність усіх архітектурних модулів платформи. Інтерфейс користувача інтуїтивно зрозумілий і коректно реагує на дії. Інтеграція із зовнішніми сервісами працює стабільно, без затримок обробляючи JSON-відповіді, що гарантує безшовний користувацький досвід.

4.4 Впровадження системи та розгортання

Процес впровадження розробленої платформи «BookSharing» включає підготовку інфраструктури, налаштування середовища виконання та

ініціалізацію бази даних. На даному етапі життєвого циклу проєкту розгортання здійснюється у тестовому середовищі, що дозволяє провести фінальну демонстрацію функціонала перед потенційним виведенням продукту на етап комерційної експлуатації.

Процес розгортання серверної частини складається з кількох послідовних кроків. Насамперед розгортається ізольоване віртуальне середовище Python, що гарантує відсутність конфліктів між версіями бібліотек. Після цього відбувається автоматизоване встановлення всіх необхідних залежностей за допомогою менеджера пакетів «pip» на основі файлу специфікацій requirements.txt.

Важливим етапом розгортання є конфігурація змінних оточення. З міркувань безпеки, чутливі дані не зберігаються у вихідному коді. Усі криптографічні ключі доступу, зокрема SECRET_KEY фреймворку Django, ключі для інтеграції з Google Books API та токени авторизації для логістичного шлюзу «Нової Пошти», винесені в окремий конфігураційний файл .env.

Ініціалізація структури бази даних здійснюється за допомогою вбудованої системи міграцій Django. Виконання особливих команд дозволяє системі автоматично згенерувати SQL-запити та створити необхідні реляційні таблиці у системі управління базами даних.

Після успішного застосування міграцій запускається локальний WSGI-сервер розробки. Платформа стає доступною для тестування та взаємодії через веббраузер за стандартною адресою. Такий підхід до розгортання є оптимальним для етапу тестування, оскільки забезпечує швидкий зворотний зв'язок та надає розширений інструментарій для відстеження помилок і мережевих запитів.

4.5 Оцінка продуктивності та перспективи масштабування

Для об’єктивної оцінки якості розробленого вебзастосунка, його швидкодії та відповідності сучасним стандартам веброзробки було проведено комплексний аудит за допомогою інструмента Google Lighthouse. За результатами фінального тестування система продемонструвала високі показники за всіма ключовими

напрямками (рис. 4.4). Зокрема, загальна ефективність склала 96 балів, використання оптимальних методів розробки отримало максимальні 100 балів, рівень пошукової оптимізації оцінено у 91 бал, а доступність інтерфейсу – у 82 бали.

Особливу увагу під час аудиту було приділено показникам Core Web Vitals, які безпосередньо впливають на користувацький досвід. Усі ключові метрики мають нормальні показники, час до появи першого візуального елемента становить лише 0,6 секунди, час відтворення найбільшого видимого елемента – 1,4 секунди, а повна відсутність блокування основного гарантує миттєву реакцію сторінки на дії користувача. Таких високих показників швидкодії вдалося досягти завдяки практичній оптимізації ресурсів на стороні клієнта. Під час первинного аналізу було виявлено затримки завантаження через неоптимальне використання графіки, зокрема завантаження файлу логотипу з надлишковою роздільною здатністю. Зміна його розміру відповідно до фактичних в інтерфейсі дозволив суттєво зменшити обсяг мережевого трафіку та підвищити фінальну оцінку продуктивності додатка.

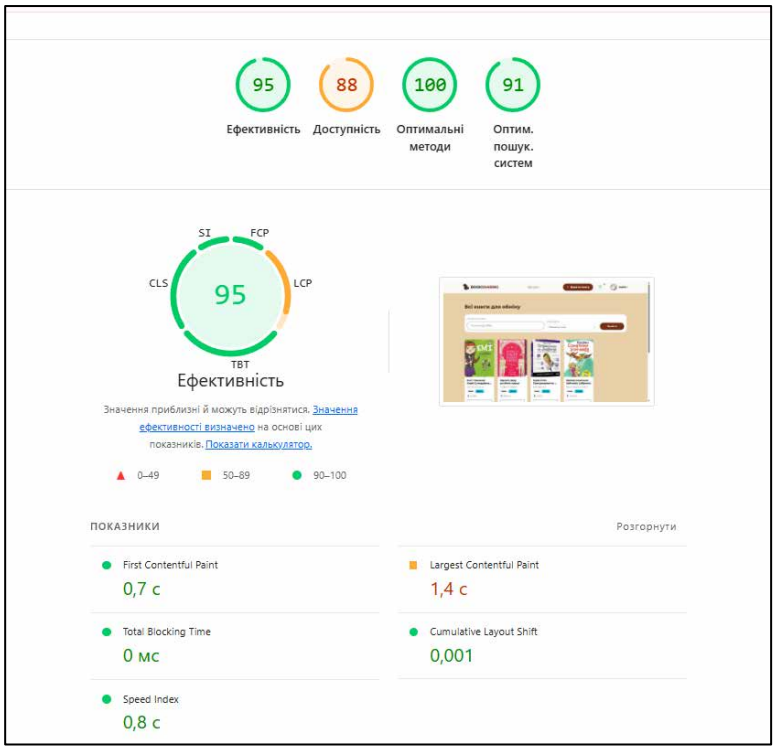

Рис. 4.4 Результати тестування продуктивності додатка за допомогою Google Lighthouse

Наразі поточна архітектура вебзастосунка повністю покриває наявні вимоги щодо швидкодії, проте для забезпечення стабільної роботи в умовах збільшення кількості активних користувачів та обсягів даних розроблено стратегію подальшого масштабування. При досягненні показників критичного навантаження першочерговим кроком стане вертикальне масштабування, що передбачає планомірне збільшення апаратних потужностей сервера. Разом із цим передбачається подальша оптимізація роботи бази даних шляхом профілювання складних запитів, налаштування додаткових індексів та, за необхідності, фізичного винесення СУБД на окремий виділений сервер для розмежування обчислювальних ресурсів. На рівні програмного коду зниження навантаження забезпечуватиметься завдяки впровадженню алгоритмів локального кешування для збереження результатів найбільш об’ємних операцій, а також переходу до асинхронної обробки тривалих фонових завдань. Крім того, для швидкої доставки статичного контенту доцільним кроком стане підключення мережі доставки контенту, що додатково розвантажить основний сервер та гарантуватиме високу відмовостійкість і стабільність вебзастосунка на всіх етапах його життєвого циклу.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно вирішено актуальне науково-практичне завдання, яке полягає у комплексному дослідженні, проєктуванні та розробці сучасного вебзастосунка. На початковому етапі дослідження було проведено глибокий аналіз предметної області та огляд існуючих програмних аналогів на ринку. Це дозволило виявити їхні основні недоліки, визначити ключові потреби цільової аудиторії та сформувати детальні функціональні й нефункціональні вимоги до майбутньої системи. Обґрунтовано актуальність створення власного продукту, який би поєднував високу продуктивність, зручний інтерфейс та надійність обробки інформації. Завдяки всебічному аналізу було обрано оптимальний стек технологій, що забезпечує ідеальний баланс між швидкістю розробки, безпекою та можливістю подальшого масштабування системи.

На етапі системного аналізу та проєктування було розроблено загальну архітектуру програмного рішення. Створено концептуальну та логічну моделі даних, що лягли в основу проєктування нормалізованої реляційної бази даних. Така оптимізована структура гарантує високу цілісність інформації, усуває надлишковість даних та створює надійне підґрунтя для швидкого виконання складних пошукових запитів. Крім того, було спроєктовано базові користувацькі сценарії та розроблено прототипи інтерфейсу, що дозволило заздалегідь продумати логіку взаємодії користувача з системою та забезпечити високий рівень ергономіки.

Практична реалізація проєкту охопила розробку повноцінної інтегрованої серверної та клієнтської частин. Серверний модуль був реалізований для ефективної обробки бізнес-логіки, безпечної автентифікації користувачів, управління сесіями та захищеної взаємодії з базою даних. Одночасно з цим розроблено сучасний клієнтський інтерфейс. Використання передових підходів до адаптивної верстки гарантує коректне відображення та збереження повного

функціоналу системи на пристроях із різними розмірами екранів, включаючи персональні комп'ютери, планшети та смартфони, що є критично важливою вимогою для сучасного програмного забезпечення.

На завершальному етапі дослідження було проведено комплексне тестування, оптимізацію та оцінку продуктивності готового рішення. Аудит за допомогою інструмента Google Lighthouse продемонстрував високу ефективність додатка. З огляду на потенційне зростання популярності додатка та збільшення обсягів інформації, було сформовано стратегію його подальшого масштабування. Цей план охоплює подальшу оптимізацію запитів, впровадження механізмів локального кешування та вертикальне розширення апаратних ресурсів сервера. Отже, розроблений вебзастосунок є повністю функціонально завершеним і стабільним продуктом, який досяг поставленої на початку дослідження мети, відповідає всім висунутим вимогам і повністю готовий до практичної експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wiegers K., Beatty J. Software Requirements. 3rd ed. Microsoft Press, 2013. 672 p.
2. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2020. 1166 p.
3. Burkov A. The Hundred-Page Machine Learning Book. Andriy Burkov, 2019. 160 p.
4. Офіційна документація Scikit-learn: Machine Learning in Python. [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/ – Назва з екрана. – Дата звернення: 10.05.2026.
5. Офіційна документація фреймворку Django. Django Software Foundation. [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/ – Назва з екрана. – Дата звернення: 02.03.2026.
6. Офіційна документація PostgreSQL. The PostgreSQL Global Development Group. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ – Назва з екрана. – Дата звернення: 15.03.2026.
7. OAuth 2.0. OAuth.net. [Електронний ресурс]. – Режим доступу: https://oauth.net/2/. – Назва з екрана. – Дата звернення: 04.04.2026.
8. Chen P. P.-S. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*. 1976. Vol. 1, No. 1. P. 9–36.
9. Object Management Group. About the Unified Modeling Language Specification Version 2.5.1. [Електронний ресурс]. – Режим доступу: https://www.omg.org/spec/UML/ – Назва з екрана. – Дата звернення: 20.04.2026.

10. Показники книжкової галузі України 2024. Kyiv Book Fest : офіційний сайт. [Електронний ресурс]. – Режим доступу: https://kbf.org.ua/news/pokazniki-knizhkovoi-galuzi-ukraini-2024 – Назва з екрана. – Дата звернення: 10.05.2026.
11. Грицюк Ю. І. Проєктування програмного забезпечення : навчальний посібник. Львів : Вид-во ЛДУ БЖД, 2018. 320 с.
12. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ: Видавнича група BHV, 2017. 384 с.
13. Офіційна документація фреймворку Pytest. Pytest : офіційний сайт. [Електронний ресурс]. – Режим доступу: https://docs.pytest.org/en/latest/ – Назва з екрана. – Дата звернення: 04.05.2026.
14. Офіційна документація API Нової Пошти. Нова Пошта : портал розробника. [Електронний ресурс]. – Режим доступу: https://developers.novaposhta.ua/ – Назва з екрана. – Дата звернення: 18.04.2026.
15. Karl Wiegers Software Development Pearls: Lessons from Fifty Years of Software Experience, 2021
16. Google Books APIs Documentation. Google for Developers. [Електронний ресурс]. – Режим доступу:https://developers.google.com/books/docs/overview – Назва з екрана. – Дата звернення: 25.04.2026.
17. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge : Cambridge University Press, 2008. 496 p.

ДОДАТОК А

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Фастовець Анастасія Валеріївна, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Веб-орієнтована інформаційна система обліку книжкових ресурсів між користувачами» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що інструмент ШІ Gemini були використані для:

- перекладу іншомовних джерел;
- стилістичного редагування та перевірки граматики;
- допомоги у написанні/відлагодженні програмного коду;

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«13» травня 2026 р. _____ **Анастасія ФАСТОВЕЦЬ**