Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ПЛАТФОРМИ ОБ'ЄДНАНОГО ПОШУКУ АВТОРСЬКИХ ТЕКСТІВ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1
Верепа Д. С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О. А.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Верепі Денису Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка платформи об'єднаного пошуку авторських текстів

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 31 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека машинного навчання ML.NET.

4. Перелік питань, що потрібно опрацювати в роботі
1. Актуальність та переваги об'єднаного пошуку авторських текстів.
2. Розробка алгоритму отримання та зберігання даних книг з оригінальних джерел.
3. Проєктування алгоритму машинного навчання контентно-орієнтованих рекомендацій.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, постановка задачі, схеми роботи платформи, діаграма за стандартом ERD, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-13.04.25	
4	Аналіз технічних засобів	14.04.25-15.04.25	
5	Розробка методу	16.04.25-25.04.25	
6	Програмна реалізація	26.04.25-15.05.25	
7	Оформлення пояснювальної записки	16.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-08.06.25	
12	Занесення роботи в електронний архів	02.06.25-08.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-09.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Кобилін О. А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 76 с., 4 табл., 38 рис., 36 джерел.

BLAZOR, ASP.NET CORE, REST API, ВЕБСКРАПІНГ, МАШИННЕ НАВЧАННЯ, SQL SERVER, ENTITY FRAMEWORK CORE, СИСТЕМИ РЕКОМЕНДАЦІЙ, ПРОКСІ-СЕРВІС.

Об’єктом роботи є пошук та рекомендації літературних творів у відкритих інтернет-джерелах.

Метою роботи є розробка єдиної вебплатформи, що об’єднує популярні ресурси з безкоштовними авторськими текстами й надає результати пошуку та персональні рекомендації без потреби переходу на кожен вебсайт окремо.

Використано принципи об’єктно-орієнтованого проєктування клієнт-серверних REST-систем, методи вебскрапінгу для збирання метаданих творів. Реалізовано контентно-орієнтований підхід до формування персональних рекомендацій на основі аналізу описів, тегів і числової статистики.

У результаті роботи здійснена програмна реалізація платформи для об’єднаного пошуку і рекомендацій авторських текстів з різних оригінальних джерел.

BLAZOR, ASP.NET CORE, REST API, WEB SCRAPING, MACHINE LEARNING, SQL SERVER, ENTITY FRAMEWORK CORE, RECOMMENDER SYSTEMS, PROXY-SERVICE.

The object of the work is the search and recommendation of literary works from open Internet sources.

The aim of the work is to develop a unified web platform that aggregates popular sources of free copyright texts and provides search results and personalized recommendations without requiring users to visit each website individually.

The project applies principles of object-oriented design for client-server REST systems and employs web-scraping methods to collect metadata about the works. A content-based approach to generating personalized recommendations is implemented through analysis of descriptions, tags, and numerical statistics.

As a result, a software platform has been implemented that offers unified search and recommendations for authorial texts drawn from various original sources.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DI – Dependency Injection (впровадження залежностей)

БД – база даних

SQL − Structured Query Language (мова структурованих запитів)

LINQ – Language Integrated Query (мова запитів до колекцій)

JSON – JavaScript Object Notation (формат обміну даними між клієнтом і сервером)

API – Application Programming Interface (інтерфейс для взаємодії між різними програмними компонентами)

DTO – Data Transfer Object (об’єкт-переносник даних між шаром API та клієнтом)

SPA – Single Page Application (односторінковий вебзастосунок)

HTTP – HyperText Transfer Protocol (протокол передачі вебзапитів і відповідей)

JWT – JSON Web Token (стандарт для безпечної передачі токенів авторизації)

SignalR – бібліотека для двосторонньої реального часу комунікації між клієнтом і сервером

HTML – HyperText Markup Language (мова розмітки гіпертексту)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

UI – User Interface (інтерфейс користувача)

ВСТУП

Літературна індустрія була популярна протягом всієї історії людської цивілізації. Від дворянської еліти до звичайного сучасного члену товариства, в історії змінювалося лише кількість людей, які мали доступ до книжкового світу. Останнім сторіччям якість освіти та життя дійшли до того рівня, коли майже кожна людина на планеті може прочитати будь-який твір за своїм бажанням.

Однією з причин такого стрімкого прогресу літературного ринку є поява інтернету. Зацікавлені люди можуть ділитися своїми роботами та обговорювати деталі сюжету з читачами з усього світу.

Автори різних національностей публікують оригінальні твори на своїх регіональних вебплатформах, а перекладачі розміщують неофіційні переклади десь у своєму куточку інтернету. Окрім регіональних вебресурсів, існують глобальні гіганти, вплив яких розростає на весь англомовний світ. На таких платформах люди, незалежно від своєї національності, культури та місця проживання, діляться своїми творами, доступні та зрозумілі універсальною мовою планети.

Велика кількість подібних вебсайтів мають свою унікальну книжкову спеціалізацію, що веде до створення локальної аудиторії. Гарним прикладом для порівняння будуть три великі гіганти літературної творчості: Royal Road, Scribble Hub та Light Novel World. Один з них фокусується на перекладах іноземних азіатських новел, інший відомий за велику кількість дорослого контенту, а останній є місцем лише для оригінальних текстів. Кожен сайт має свій тип контенту та своє коло користувачів.

Актуальність теми створення розробки платформи об’єднаного пошуку авторських текстів полягає у швидкому зростанні кількості нових авторських книжок та поява нових вебресурсів з унікальним літературним профілем. Для звичайного користувача існує забагато опцій дослідження «той самого»

твору, але вони розкидані у своїх ділянках глобальної мережі та не є централізованими.

Кваліфікаційна робота має на меті вирішити проблеми прив’язки читача до великої кількості платформ та незручностей взаємодії з ними у рамках пошуку творів та зберігання читацького прогресу. Ця кваліфікаційна робота спрямована на розробку вебплатформи з метою провадити механізм пошуку літератури, що охоплює різні оригінальні вебресурси, та дати можливість створити єдиний читацький профіль, що будете відстежувати прогрес кожної книги та зберігати її у персональній бібліотеці.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА ЗАСОБІВ ДЛЯ СТВОРЕННЯ ПЛАТФОРМИ ОБ'ЄДНАНОГО ПОШУКУ АВТОРСЬКИХ ТЕКСТІВ

1.1 Актуальність та переваги об'єднаного пошуку авторських текстів

Сучасний інформаційний простір містить тисячі безкоштовних онлайн бібліотек з мільйонами електронних книжок та інших різноманітних видів літератури. Усі ці вебресурси виділяються своєю унікальною колекцією авторських творів та призначені для своєї специфічної групи читачів.

Велика кількість подібних онлайн платформ породжує проблему для звичайного користувача інтернету – знаходження потрібної книги чи тексту часто приводить до довгого окремого пошуку серед багатьох різних сайтів [1]. Такий процес забирає багато часу та зусиль, ресурси які можна було зберегти, якщо б був єдиний спосіб ефективного пошуку та візуалізації інформації.

Інша проблема, що виникає з попередньої – це неможливість одразу дізнатися який діапазон літератури мають різні джерела, – треба витрачати свій час на сканування та фільтрацію контенту.

Рішенням даної проблеми являється об'єднаний, або федеративний пошук [2]. Він здійснює сканування та збереження даних з багатьох джерел через єдиний інтерфейс. Користувачу лише потрібно ввести запит один раз, а система шукає у кількох базах даних чи бібліотек і повертає зведений список результатів.

Об'єднаний пошук буде завжди актуальним, доти існують безліченна кількість різних онлайн бібліотек. В останні роки формат електронної книги набув вибухової популярності. Світовий ринок цифрової літератури наразі перебуває у фазі стійкого зростання. За останні п'ять років обсяг продажу е-книг збільшився більш ніж на 35%, і за оцінками Future Market Insights, ринок оцінюється приблизно в 22 млн. дол. США у 2024 році [3].

Такий зріст попиту та популярності показує гарний стан та майбутнє даної індустрії взагалі. Актуальність ринку платних книжок також впливає і на сферу безкоштовних авторських творів – їх з кожним роком стає більше і більше, разом з новими вебплатформами, що зберігають їх та дають доступ читачам.

Вплив сервісів федеративного пошуку охоплюють усі задіяні сторони: читачі, автори, онлайн бібліотеки та видавці. Серед переваг для читачів можна виділити:

– економія часу та зусиль користувачів інтернету, для яких варіативність та точність вибору має первісне значення;

– виявлення творів, які інакше могли б залишитися невідомими користувачу, оскільки платформа-агрегатор може здійснювати пошук навіть у маловідомих або спеціалізованих бібліотеках;

– централізація пошуку створює основу для надання додаткових сервісів, таких як персоналізовані рекомендації творів на основі історії пошуку, відстежування читацького прогресу, створення особистої колекції, або отримування сповіщень про нові надходження;

– легкість порівняння та вибору між різними джерелами за допомогою спільного інтерфейсу, який уніфікує формат представлення результатів;

– можливість дивитися прозору статистику творів серед декількох джерел, на яких дана література опублікована.

Участь у об’єднаному пошуку для зовнішніх вебресурсів та авторів надає перевагу у вигляді додаткового трафіку та приріст нових читачів без якихось зайвих витрат на маркетинг [4]. Така система симбіотична та рідко коли конфліктує з інтересами оригінальних джерел.

Проаналізувавши переваги вебплатформ федеративного пошуку, можна зробити висновок, що актуальність розробки подібних сервісів зумовлена потребою впорядкувати та спростити доступ до величезного масиву літературного контенту в інтернеті. Подібні проєкти відповідають

сучасним тенденціям в інформаційній сфері, де на перше місце виходять універсальні інструменти пошуку та агрегатори даних, що надають користувачу релевантну інформацію максимально зручним способом.

1.2 Огляд вебресурсів, що надають доступ до електронних книг та текстів

1.2.1 Публічні цифрові бібліотеки та архіви

Масове поширення інтернету призвело до появи перших публічних електронних бібліотек, які виконували роль сховища легальної класичної літератури. Зараз такі вебсервіси накопичують тексти, що знаходяться у вільному доступі, як правило через закінчення дії авторського права або за згодою автора.

Найстарішім та найвідомішим онлайн архівом, який працює і на сьогоднішній день, являється Project Gutenberg – перша у світі цифрова бібліотека, заснована Майклом Хартом у 1971 році [5]. Станом на 2025 рік проєкт надає більше 75 тисяч безкоштовних повнотекстових творів у форматах EPUB, Kindle, HTML та TXT. Це переважно класична література, що перейшла у суспільне надбання, а також деякі інші документи. Інтерфейс сайту (рис. 1.1) дозволяє шукати книги за назвою або автором, переглядати тематичні каталоги та завантажувати файли в різних форматах.

Іншим відомим вебресурсом є Internet Archive (IA), також відомий під роллю «всесвітньої бібліотеки». На відміну від Project Gutenberg, дана вебплатформа зберігає більш різноманітний контент – книги, аудіо, відео та копії вебсторінок. Підпроєкт Open Library має близько 2 млн. сучасних книг, а загалом колекція містить близько 28 млн. текстів, включно з оцифрованими книгами публічного домену, журналами та документами [6].

Ще одним прикладом являється Wikisource, бібліотека першоджерел, де волонтери викладають тексти класичних творів різними мовами.

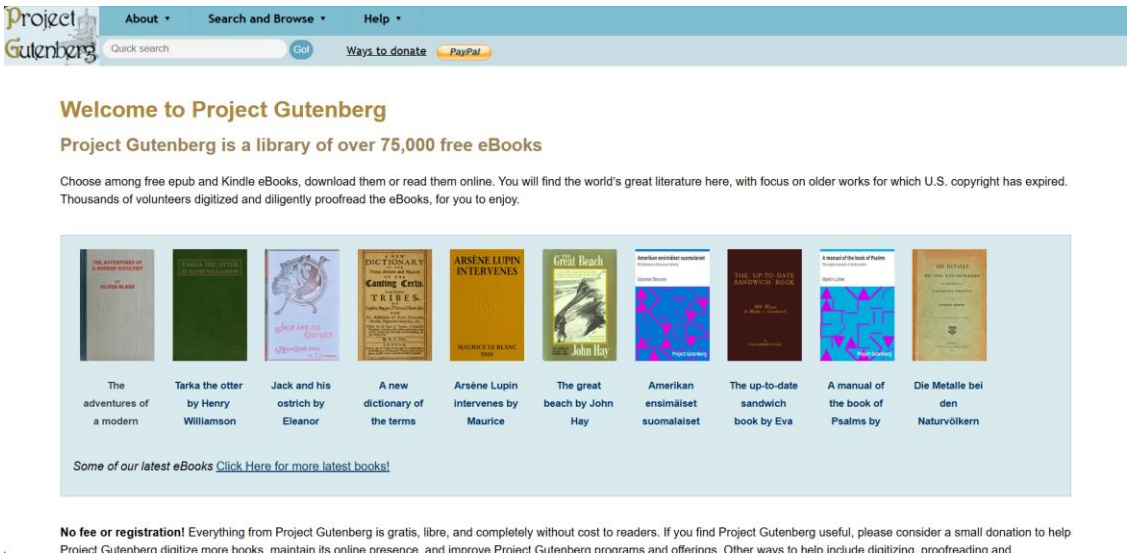

Рисунок 1.1 – Головна сторінка найстарішої е-бібліотеки Project Gutenberg [5]

Проєкт є повністю вільною та некомерційною онлайн бібліотекою, містить літературу у публічному домені або під вільними ліцензіями. На сьогоднішній день вебплатформа охоплює 79 мов та зберігає понад 6,4 млн. сторінок-текстів. Доступ до такої інформації не обмежується реєстрацією або іншим механізмом.

Серед українського сегменту публічних бібліотек виділяються три представники – «Відкрита книга», «Чтиво» та «УкрЛіб». Останній вебресурс позиціонується як найбільша україномовна електронна бібліотека і містить приблизно 60 тис. текстів: шкільна програма, класика, біографії тощо.

Усі ці ресурси працюють за однаковим принципом: публікують лише твори, що перейшли в суспільне надбання, або ті, які автори дозволили розповсюджувати безкоштовно.

1.2.2 Онлайн-бібліотеки сучасної літератури

Основна відмінна онлайн бібліотек сучасної літератури від публічних архівів – це робота з чинним авторським правом. Багато таких вебплатформ

працюють на комерційній основі, проте інколи мають розділи з безкоштовними матеріалами.

Найбільш популярними літературними вебресурсами являються такі світові гіганти:

– Amazon Kindle Store, один з найкрупніших учасників ринку електронних книжок, має понад 44 млн. творів різними мовами. Частина з них доступна безкоштовно для читання під час акцій чи за програмою Kindle Unlimited , однак переважно контент пропонується за моделлю продажу;

– Google Books – інший ключовий гравець, зберігаючий та представляючий понад 40 млн. відсканованих книг. Через вебінтерфейс Google Books користувач може шукати фрагменти тексту у книгах і переглядати уривки. Повний перегляд чи скасування доступні лише для деяких видань (що в публічному домені або надані видавцями). Також Google пропонує платформу Play Books для придбання е-книг та можливість завантажити безкоштовні класичні твори;

– Barnes & Noble – найбільша книжкова мережа США – також підтримує власний онлайн-магазин eBook (Nook), де налічується понад 4,5 млн. електронних книг.

Комерційні онлайн бібліотеки сучасної літератури часто дозволяють читати книги прямо на сайті або завантажувати їх для офлайн-прочитання. Доступ до більшості електронних книг вимагає від користувача реєстрації та виконання транзакції покупки товару. Безкоштовний контент надходить у вигляді класичних творів або промо-видання від авторів і видавців.

1.2.3 Платформи самовидавництва та веброманів

Своєрідним логічним продовженням сучасних онлайн-бібліотек стали ті вебмайданчики, де автор уже не просто обирає «де викласти готову книжку», а безпосередньо творить її «на очах» у читача, паралельно

отримуючи статистику, коментарі й навіть донати. Саме тут зароджується новітня форма літературної екосистеми – серіалізовані вебромани, що публікуються главами, а їхня популярність рахується не тиражами, а лайками, підписками та хвилинами прокруту.

Wattpad нині лишається наймасовішим осередком для аматорських романів – лише у лютому 2025 року сайт зібрав 84,2 млн. відвідувань при середній тривалості сеансу майже дев'ятнадцять хвилин, причому кожен гість гортає понад дванадцять сторінок за візит [7]. Левова частка контенту й досі безкоштовна й демонструється за підтримки реклами, водночас існує підписка на Wattpad Premium Picks (рис. 1.2) для доступу до ексклюзивних книг без оголошень, а також можливість купувати окремі розділи через Wattpad Paid Stories. Соціальна взаємодія (коментарі «построчно», репости у TikTok-fanvid формі) перетворює читання на колективну гру.

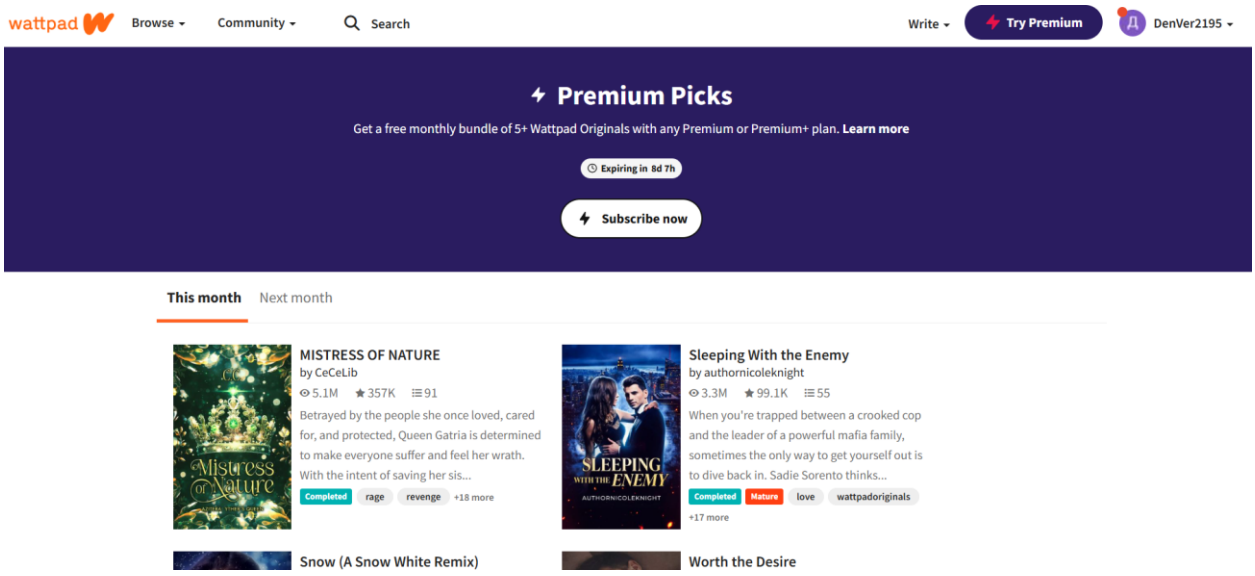

Рисунок 1.2 – Сторінка Wattpad Premium Picks [8]

Royal Road, що позиціює себе як «writer-friendly community», за той самий період набрало 13,9 млн. сесій [7]. Статистично це майже вшестеро менше за Wattpad, однак платформа приваблює аудиторію більш «геймерським» підходом: детальна система тегів, щоденні рейтинги, «покроковий» прогрес написання та можливість швидко прикріпити Patreon.

Хоча більшість глав відкриті для безкоштовного читання, підтримати автора можна через зовнішні донати (Patreon, Ko-fi) або внутрішній фан-клуб за підпискою, що стимулює регулярну підтримку творчості.

Scribble Hub, тісно пов’язаний із форумною культурою, слугує своєрідним інкубатором нішевих жанрів – зокрема гендер-бендер фентезі та slice-of-life ісекаю. Хоч щомісячна відвідуваність усього 4,5 млн. користувачів [9], саме тут нерідко з’являються твори, що потім набирають обертів на старших майданчиках. Усі глави доступні безкоштовно. За бажанням користувач може пожертвувати авторам через PayPal чи Patreon.

Light Novel World показує, що агрегація перекладених азійських ранобе теж може стати бізнес-моделлю: 7,9 млн. візитів за лютий та середній час перебування понад вісімнадцять хвилин. Попит у першу чергу генерують фанати корейських та китайських фентезі-серіалів, котрі шукають швидкі неофіційні переклади. Звідси й парадокс – платформа існує на межі, водночас стимулюючи попит і провокуючи дискусії про авторське право. Усі переклади відкриті без плати, але платформа заробляє на рекламі та донатах від користувачів, створюючи атмосферу краудфандингу для швидких неофіційних перекладів.

WuxiaWorld, вебсайт, який називають піонером «легальних» перекладів китайських xianxia-епопей. Попри «лише» 3,7 млн. відвідувань [10], майданчик запровадив модель раннього доступу через Karma Shop (рис. 1.3), де користувач купує віртуальну валюту, щоб читати найновіші глави до офіційного релізу, щоб просто читати глави коли безкоштовний період закінчується, або накопичує бали в рамках щоденних активностей. Платформа активно купує ліцензії у китайських видавців, а після перекладу офіційні томи виходять друком у США.

Не можна оминути й Tapas – англомовний філіал корейського Kakao, який після реструктуризації 2024 року зробив ставку на конвертацію популярних веброманів у webtoon-формат і вже мріє про власне IPO (Initial Public Offering) [11]. Більшість перших розділів доступні безкоштовно, а

подальше читання вимагає оплати за допомогою внутрішніх монет або оформлення підписки Tapas Premium; реклама при цьому мінімізується для підписників. Їхній кейс показує, що майбутнє ринку бачиться у «контентному ланцюжку», де прозовий серіал стає коміксом, а далі – аніме чи грою, і кожний етап підсилює попередній.

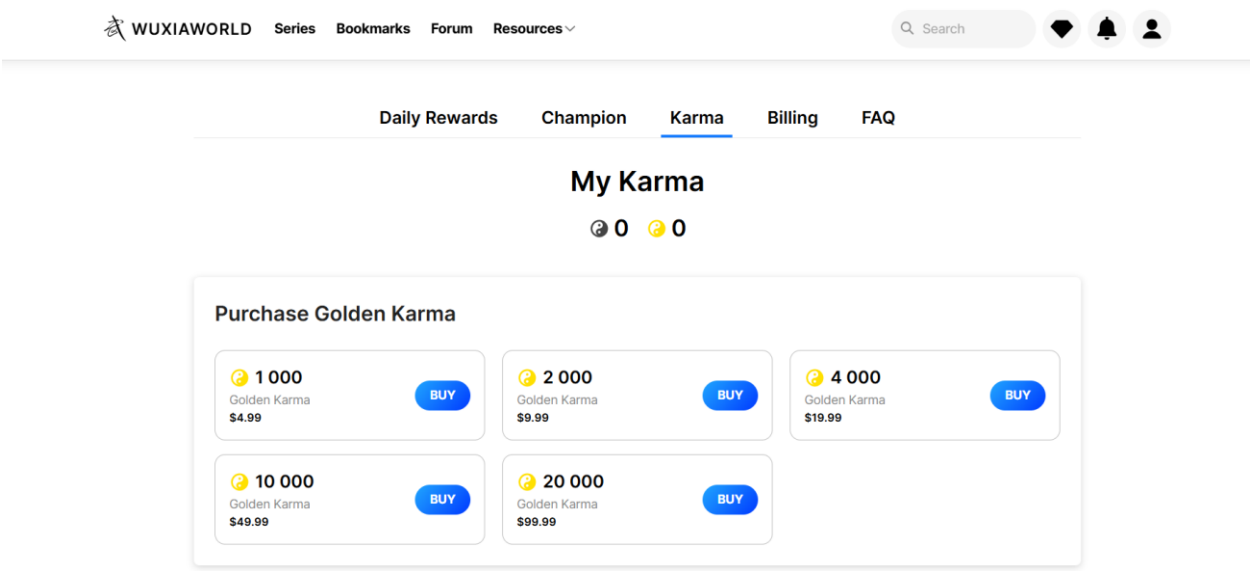

Рисунок 1.3 – Сторінка Karma Shop вебсайту WuxiaWorld [12]

Таким чином, платформи самвидавництва формують гібридну екосистему, де безкоштовні модель з рекламою та донатами співіснують із передплатою й мікротранзакціями. У таких умовах для ad-driven сервісів ключовими є метрики глибини прокруту, часу сесії та відвідуваності, тоді як для платних моделей важливі коефіцієнт конверсії в покупку, кількість активних підписок і обсяг внутрішніх мікротранзакцій. Завдяки цьому читач стає не просто споживачем, а активним учасником «літературного ланцюжка», а автор – водночас маркетологом і менеджером спільноти.

1.3 Порівняння існуючих систем-агрегаторів та аналіз принципів роботи проксі-платформ

1.3.1 Механізм федеративного пошуку в системах-агрегаторах

Агрегаторні системи пошуку авторських текстів забезпечують єдиний інтерфейс для доступу до даних з багатьох бібліотечних джерел [2]. Вони працюють за принципом федеративного пошуку: користувач формулює один запит, а система надсилає його до кількох баз даних або сервісів і отримує об’єднаний список результатів (рис. 1.4).

Це дозволяє суттєво заощадити час на пошук книги в різних каталогах і веббібліотеках [1]. Наприклад, якщо раніше потрібно було окремо шукати твір в Open Library, Google Books чи на інших сайтах, то агрегатор здатен паралельно опитати всі ці джерела і видати зведені результати у одному вікні. Таким чином, досягається централізація пошуку та уніфікація форматів представлення інформації, що полегшує порівняння різних копій або видань твору.

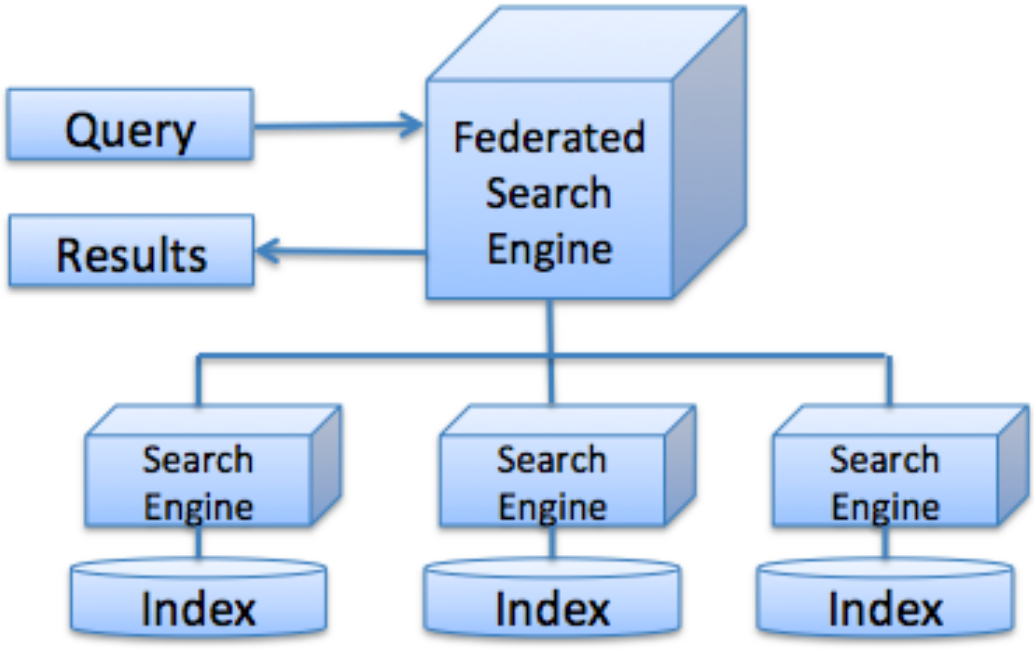

Рисунок 1.4 – Схема роботи федеративного пошуку [2]

1.3.2 Офіційні платформи-агрегатори та їхні бізнес-моделі

Одним із найстаріших прикладів агрегатор-платформ можна виділити вебсервіс Open Library. Своє основне завдання платформа описує як «одна сторінка – одна книжка», вона збирає метадані з бібліотечних каталогів та фондів Internet Archive, не лякаючись «ручної» волонтерської правки. Станом на 2025 рік картотека являється наймасштабнішим відкритим каталогом в інтернеті, зберігаючи понад 30 млн. записів: від повних оцифрованих текстів до лише обкладинок й ISBN. Сервіс API є дуже простим, дозволяючи будь-якому застосунку легко підтягнути назву книги, автора чи прев'ю. Єдиною проблемою являється вплив правовласників – більшість сучасних книжок видно лише у вигляді карток, читати можна хіба невеличкі уривки.

Комерційну нішу заповнив платний сервіс ISBNdb, який торгує не файлами, а метаданими. Чотири десятки мільйонів назв, дев'ятнадцять полів у відповіді, ще й поточні ціни з онлайн-крамниць створюють рай для вебринків. Для безкоштовного користування існує ліміт запитів та закритий код, що лишають звичайних клієнтів за бортом. Ця проблема вирішується за допомогою платної підписки.

На відміну від інших компаній, Google має трошки інший підхід: Books API об'єднує масове сканування з повнотекстовим пошуком, що пролізе по заданій фразі крізь сорок-плюс мільйонів PDF і покаже конкретну сторінку, а іноді дозволить подивитися цілу книгу прямо у браузері. Купити примірник можна у сервісі «Play Books». Даний сервіс має схожу проблему з Open Library – повністю відкриті твори лише публічного домену, решта надає лише уривок, або й зовсім «No preview».

1.3.3 Нелегальні агрегатори у децентралізованих мережах

На противагу офіціозу виросла Library Genesis – децентралізований «спрут» на десятки дзеркал. У базі, за останньою оцінкою спільноти, приблизно 2,4 млн. наукових та 2,2 млн. художніх книжок, плюс журнали й комікси [13]. Каталог аскетичний: пошук, посилання, кнопка «Download». Перевага очевидна – нуль бар'єрів; недолік теж – постійні блокування, різношерсті метадані та необхідність шукати живий домен.

З LibGen виросла гілка Z-Library. Ті самі файли, зате приємніший інтерфейс, персональні полиці та рекомендації. За рахунок донатів і реєстрацій платформа стримує навантаження: без акаунту день-два і ліміт вичерпався. 2022-го домен зняли, тож команда перейшла на схему SingleLogin: користувач отримує власний піддомен-проксі, а ще можна піти в Tor. Звідси плюси – мільйони книг і «домашня» зручність; мінуси – фішингові копії, регулярна міграція URL і повна нестабільність з точки зору закону.

Для забезпечення стійкого доступу до згаданих тіньових бібліотек з'являються метапошуки на кшталт Anna's Archive, які індексують LibGen, Z-Lib та Sci-Hub і віддають єдиний результат навіть тоді, коли чергове дзеркало вже впало [14].

1.3.4 Порівняння агрегаторних і проксі-платформ

У таблиці 1.1 зведено основні характеристики розглянутих платформ-агрегаторів і проксі-рішень для пошуку книг та авторських текстів:

Як видно з таблиці, агрегаторні системи суттєво відрізняються за природою. Легальні платформи (Open Library, Google Books) працюють із правовласниками, мають API й віджети, гарантують стабільність, але обмежені рамками авторського права. З іншого боку, тіньові сервіси (LibGen,

Z-Lib) прагнуть повного доступу без бар'єрів – їх колекції значно більші, проте вони порушують закон.

Таблиця 1.1 – Порівняльна характеристика агрегаторних та проксі-платформ для пошуку книг

Платформа	**Тип**	**Обсяг**	**Переваги**	**Недоліки**
Open Library	Відкритий агрегатор; каталог (легальний)	~30 млн. записів; дані з бібліотечних каталогів і Internet Archive	Відкритий API, читання онлайн, спільнота волонтерів	Багато сучасних книг лише як метадані (немає повного тексту)
ISBNdb	Комерційна агрегатор-БД (легальна)	43+ млн. назв; метадані від видавців, ISBN-реєстрів тощо	Детальна інформація про книги, включно з цінами; стабільний API	Платний доступ, недоступний широкому загалу без API
Google Books	Агрегатор та бібліотека	40+ млн. книг	Повнотекстовий пошук по книгах	Обмежений перегляд
Library Genesis	Тіньова бібліотека-агрегатор (нелегальна)	~4,6 млн. книг (наукових і художніх) + журнали, статті	Величезна кількість безкоштовних текстів	Незаконний статус; домени часто блокуються

Продовження таблиці 1.1

Платформа	**Тип**	**Обсяг**	**Переваги**	**Недоліки**
Z-Library	Тіньова бібліотека; вебплатформа з проксі-доменами	13+ млн. книг, 84+ млн. статей; першоджерело – LibGen + внесок користувачів	Зручний інтерфейс, персональні полиці, рекомендації; багато форматів (PDF, EPUB, MOBI тощо)	Незаконний статус; вимога реєстрації для більшості функцій; ризик фішингових дзеркал
Anna’s Archive	Метапошук / агрегатор (нелегальний контент)	40 млн. книг, 98 млн. статей (індекс); джерела – LibGen, Z-Lib, Sci-Hub, тощо	Єдиний пошук по кількох базах; відкритий код і дані; стійкість через IPFS/торренти	Не хостить власних файлів; можливі дублі в результатах

Проксі-платформи (дзеркала Z-Lib, Anna’s Archive) виникли як відповідь на цензуру й блокування. Вони дублюють і розподіляють дані, підвищуючи живучість контенту, від простого перенаправлення запитів до розподілених мереж (IPFS). У сукупності офіційні агрегатори та проксі формують екосистему, де зручний пошук поєднується зі стійкістю доступу.

1.4 Інструментарій та фреймворки для створення вебзастосунку на основі .NET (Blazor Web App)

1.4.1 Платформа .NET та Blazor: огляд і сучасний стан

Платформа .NET – це відкрита кросплатформна середа з керованим виконанням, що дозволяє розробляти додатки для Windows, Linux та macOS за допомогою єдиного набору мов і бібліотек (.NET 5+ об’єднує колишні .NET Framework і .NET Core). Модель випуску SDK передбачає щорічні релізи, серед яких позначаються як LTS-версії з довгостроковою підтримкою, так і менш тривалі релізи з найсвіжішими фічами. Для веброзробки на базі .NET ключовим компонентом є ASP .NET Core – легковага, модульна та розширювана платформа, що підтримує побудову REST-API, MVC-додатків, мікросервісів і серверного рендерінгу.

У складі ASP .NET Core представлений Blazor – фреймворк компонентного підходу, який дає змогу описувати інтерфейс і логіку на C# та Razor-шаблонах замість JavaScript. Завдяки цьому можна використовувати один технологічний стек для фронтенду й бекенду, повторно застосовувати модулі та бібліотеки, а також об’єднати середовища розробки.

Blazor підтримує два основні режими хостингу: клієнтський та серверний.

Blazor WebAssembly (WASM): клієнтський застосунок завантажується в браузер у вигляді WebAssembly-модуля разом із .NET-рантаймом та збірками [15]. Усі UI-обчислення й обробка подій виконуються локально, а взаємодія з сервером відбувається через API-виклики. Це забезпечує:

– автономну роботу після початкового завантаження (можливість офлайн-режиму);

– зниження навантаження на сервер;

– швидкий відгук інтерфейсу.

Водночас до недоліків належать більший початковий розмір завантаження і залежність від підтримки WebAssembly у браузері.

Blazor Server: весь застосунок працює на сервері в контексті ASP .NET Core, а клієнту передається тільки мінімальний HTML та диф-фрагменти оновлень через двосторонню WebSocket-сесію на базі SignalR. Це дає такі переваги:

– компактний клієнт із мінімальними залежностями;

– централізоване зберігання коду й даних, що спрощує питання безпеки та розгортання;

– краща сумісність зі старими браузерами.

Основні обмеження – потреба в постійно якісному з'єднанні та затримки мережі, які безпосередньо впливають на відгук UI [15].

Вибір між моделями хостингу слід робити, орієнтуючись на вимоги до:

– автономності та офлайн-доступу;

– початкового часу завантаження;

– розподілу навантаження між клієнтом і сервером;

– гарантій безперервності роботи при перебоях мережі.

Загалом, .NET із Blazor пропонує потужний і гнучкий інструментарій для створення сучасних SPA та гібридних веб-додатків, дозволяючи адаптувати архітектуру під будь-які завдання й сценарії.

1.4.2 Інструменти розробника та компонентний підхід

Розробка вебзастосунків на базі .NET найчастіше здійснюється в середовищі Microsoft Visual Studio, яке надає комплексний набір засобів для роботи з Blazor-проєктами [16]. Вбудовані шаблони створення проєктів, підсвічування синтаксису та IntelliSense полегшують написання коду, а потужний відладчик і можливість Hot Reload дозволяють миттєво вносити зміни в C# або Razor і відразу бачити результат у браузері без повної перекомпіляції. Інтеграція з Git, засоби профілювання й модульного тестування, а також можливість розгортання застосунку напряму на сервер

чи в хмару забезпечують злагоджений робочий процес і дозволяють зосередитися на бізнес-логіці замість рутинних операцій.

Blazor реалізує компонентний підхід, де будь-який елемент інтерфейсу визначається як Razor-компонент – єдиний блок UI, який поєднує розмітку HTML із логікою на C#. Компоненти приймають параметри, обробляють події та можуть вкладати один одного, формуючи ієрархію повторно використовуваних елементів. Однонаправлений потік даних гарантує, що оновлення властивостей у батьківському компоненті коректно передаються до дочірніх, а події з дочірніх рівнів піднімаються вгору для обробки. При зміні стану компонента Blazor виконує перерахунок віртуальної DOM і оновлює лише ті частини інтерфейсу, які справді зазнали змін, що сприяє економії ресурсів і підвищенню продуктивності. Такий підхід упорядковує кодову базу, полегшує тестування та повторне використання компонентів незалежно від конкретного проєкту.

1.4.3 Робота з даними, ML.NET та супутні служби (EF Core, Identity, SignalR, Azure)

Для зберігання й обробки даних у Blazor Web App використовується Entity Framework Core – ORM, що через контекст (DbContext) і набори сутностей (DbSet) забезпечує прозору роботу з різними СКБД (SQL Server, SQLite, PostgreSQL тощо), міграції схеми та транзакції без написання сирих SQL-запитів. Контекст реєструється в DI-контейнері ASP .NET Core, що дає змогу виконувати LINQ-запити в службах, API-контролерах або Razor-сторінках і повертати результати у форматі JSON клієнту для динамічного відображення.

Для розширення можливостей роботи з даними та впровадження інтелектуальних сценаріїв у застосунку зазвичай задіюють дві ключові технології:

– EF Core для ORM-шару, який керує моделлю об’єктів, міграціями та транзакціями;

– ML .NET для побудови й інтеграції моделей машинного навчання (класифікація тексту, NLP-задачі, рекомендаційні системи тощо) без виходу за межі .NET, із підтримкою TensorFlow, ONNX та готових високорівневих API (MLContext, DataView, Transformers, Trainers).

Крім базового пошуку в базі через EF Core, ML .NET дає змогу аналізувати схожість документів, класифікувати запити й підвищувати релевантність результатів. Модель просто інкапсулюється в ASP .NET Core-сервісі і за необхідності запускається на сервері або навіть у WebAssembly-клієнті (за умови компактності й простоти обчислень).

ML .NET фактично привносить у традиційну архітектуру «дані-логіка-UI» четвертий вимір – здатність застосунку навчатися на власному досвіді. Це переводить веб-сервіс із пасивного реагування на запити у режим проактивного передбачення потреб користувача, формуючи персоналізований досвід без зайвого ускладнення коду для розробника [17].

Для забезпечення безпеки, комунікації в реальному часі та хмарного розгортання використовують ще три важливі компоненти:

– ASP .NET Core Identity для управління користувачами, ролями та аутентифікацією (cookie або JWT);

– SignalR для двосторонньої комунікації в реальному часі (сповіщення про нові дані, реактивні оновлення списків тощо);

– Azure (App Service для хостингу, Azure SQL або Cosmos DB для зберігання даних, Azure Cognitive Services і Azure SignalR Service для масштабованих AI- та real-time-завдань).

Такий набір інструментів і фреймворків забезпечує повний стек для будь-яких вебзастосунків на .NET: від роботи з даними й аналітики до безпеки, інтерактивності й хмарної масштабованості.

1.5 Постановка задачі

Станом на сьогоднішній день, вебплатформи-агрегатори та вебсервіси федеративного пошуку, що спеціалізуються на літературному контенті, являються актуальними інструментами для більш серйозного читача, який готовий потратити свій час на більш детальний та прискіпливий вибір потенційних нових творів, розкиданих серед багатьох джерел.

Більшість сервісів об’єднаного пошуку охоплюють велику кількість сторонніх бібліотек, що робить користувацьке націлювання на щось більш специфічне складною задачею. Ще одна проблема, яка корелюється з попередньою - це неможливість для подібних платформ розширити свій вплив до нішивих ресурсів, які не працюють в кооперації з агрегаторами, або не мають точок контактування взагалі – відсутність API для отримування даних.

Платформи самовидавництва та веброманів є гарним прикладом подібної некомунікативної поведінки. В них відсутні будь-який функціонал для взаємодії зі сторонніми сутностями, а те API, що є в глобальній мережі – воно зроблене на основі парсингу та скрапінгу вебсторінок і являється чисто фанатської роботою.

Вебсервіси об’єднаного пошуку також в більшості не мають функціоналу зберігання знайдених творів у своїх персональних бібліотеках, або відстежувати прогрес читання, що значно знижує привабливість таких ресурсів в очах звичайних користувачів. Потреба в єдиній, повноцінній платформі майже завжди перевищує всі переваги, що може надати простий сервіс глибокого пошуку.

Об’єктом роботи є пошук та рекомендації літературних творів у відкритих інтернет-джерелах.

Метою роботи є розробка єдиної вебплатформи, що об’єднує популярні ресурси з безкоштовними авторськими текстами й надає результати пошуку та персональні рекомендації без потреби переходу на кожен вебсайт окремо.

Для досягнення поставленої мети треба виконати наступні завдання:

– проаналізувати існуючи вебсайти самовидавництва та веброманів, системи-агрегатори пошуку текстів, для обґрунтування вибору методів об’єднаного пошуку і протоколів взаємодії з джерелами даних;

– вибрати інструментарій та фреймворки розробки на основі .NET;

– визначити архітектуру вебзастосунку (клієнт-серверна взаємодія, схема роботи проксі-сервісу) та підхід до реалізації рекомендаційного модуля;

– розробити механізм збору інформації про безкоштовні авторські тексти з декількох вибраних джерел (через API або парсинг);

– розробити механізм об’єднання та фільтрацію результатів пошуку;

– розробити систему рекомендацій літературних творів на основі аналізу поведінки користувача (історії читання, вподобань) із використанням методів машинного навчання (алгоритми ML.NET або інші моделі);

– розробити модуль користувацької бібліотеки;

– розробити браузерне розширення, що інтегрується з платформою, забезпечуючи отримання даних про читання та передавання цих даних до вебзастосунку для синхронізації прогресу між різними пристроями і ресурсами;

– розробити інтерфейс вебзастосунку за допомогою технології Blazor.

2 ПРОЄКТУВАННЯ ПЛАТФОРМИ ОБ'ЄДНАНОГО ПОШУКУ АВТОРСЬКИХ ТЕКСТІВ

2.1 Вимоги до функціоналу: пошук, рекомендації, реєстрація та особиста бібліотека

Базові, проте конче потрібні вимоги до платформи LitExplorer зібрані у всіх тих її сегментах, що відповідають за розшук літератури, видачу рекомендацій, приєднання нових читачів і керування персональною книжковою поличкою.

Мета опису – накреслити очікувану поведінкову модель системи з погляду кінцевого користувача, показавши, яким чином програмний комплекс повинен віднаходити твори, генерувати персональні підказки та забезпечувати зручний інструмент обліку прочитаного на всіх етапах, не занурюючись у технічні деталі реалізації чи приховані внутрішні схеми самої платформи.

Пошуковий модуль являється домінантною частиною інтерфейсу LitExplorer і мусить давати користувачеві шанс відкопати романи за різноманітними ознаками. Планується текстовий пошук як по повній, так і по обірваній назві твору. Додатково, до отриманих результатів дозволяється приліпити фільтри задля прояснення запиту. Серед головних фільтрів виринають такі рубрики:

– тематика й жанр (теги): читач може тикнути один чи кілька тегів із таксономії, аби натрапити на твори з відповідними рисами;

– джерело (сайт-публікатор): опція прикрутити підбірку лише до тих майданчиків, де лежать новели (скажімо, FictionPress, Wattpad та ін.);

– рейтинги: відсіювання за середнім балом твору – від мінімального і аж до найвищого можливого;

– об'єм тексту: пошук у заданому коридорі кількості розділів або неокресленої довжини;

– рік виходу: проміжок року чи навіть діапазон років публікації (якщо такі дані взагалі існують).

Видача пошуку відмальовується списком або картками творів і може переставлятися за різними критеріями. Користувачеві дозволяється обирати порядок сортування, наприклад за популярністю, середнім рейтингом, датою останнього апдейту чи чисто алфавітно, за зростанням або спаданням.

Щойно користувач міняє запит або підкручує фільтри, результати перелаштовуються на льоту, а система гарантує вивід свіжого набору записів.

Коли знайденого набігає надто багато, платформа автоматично ділить усе діло на сторінки (пагінація), аби було зручніше блукати поміж ними.

Механізм рекомендацій у LitExplorer покликаний автоматично підсовувати читачеві ті тексти, що мають добрі шанси припасти йому до смаку. Він не лише добудовує звичайний пошук, а й витягує релевантні книжки навіть без явного запиту. Рекомендації повинні бути гнучкі та суто персоніфіковані, підлаштовуючись під смаки й активність користувача.

З вимог випливає, що система підтримує два ключових режими рекомендацій:

– кураторські підказки – список творів, підібраний редакторською командою або ж алгоритмом на базі загальної популярності та рецензій (скажімо, найтрендовіші романи);

– персоналізовані підказки – добірка текстів, сформована згідно з історією читання та вподобаннями конкретного профілю (наприклад, виходячи з раніше занесених у бібліотеку романів чи виставлених тегів).

У межах очікуваної поведінки користувачеві надається перемикач між цими режимами. Рекомендовані переліки можуть демонструватися окремою сторінкою або блоками на головній чи безпосередньо в приватній бібліотеці.

Щойно профіль змінюється (додавання нового твору або правка налаштувань), персональна добірка мусить переобчислюватися, аби відкривати свіжі тексти, спираючись на інтереси читача і загальну популярність контенту.

Реєстрація з автентифікацією відповідає за персоналізацію роботи системи. Будь-який новий користувач повинен мати змогу завести обліковку зі стандартним набором даних (електронна пошта, пароль). Після успішного створення профілю відкривається доступ до власного кабінету та особистої бібліотеки. Під час наступних входів особа підтверджує себе тими самими реквізитами, а система тримає сесію відкритою аж до явного виходу. Безпечне зберігання даних і захист акаунтів – безальтернативні вимоги.

Особиста бібліотека – це приватний перелік обраних користувачем творів. Кожен зареєстрований читач може складати туди романи, що впали в око під час пошуку чи серед рекомендацій. Бібліотека дозволяє додавати й викидати позиції, а також присвоювати текстам статус («В планах», «Читаю», «Прочитано») для відстеження прогресу. Її вміст зберігається між сесіями, а записи можна упорядковувати – хоч сортуванням, хоч фільтрами за назвою, статусом або датою занесення.

2.2 Архітектура платформи та вибір програмних рішень

Архітектуру LitExplorer вигадано у дусі класичної трирівневої (N-рівневої) моделі – жорстке розділення клієнтського інтерфейсу, серверної логіки й шару зберігання даних. Така сегрегація дає змогу під кожен рівень добирати свій власний стек і залізо, майже не торкаючись сусідніх шарів.

Через це й поява нових романів, і радикальна зміна структури метаданих зачіпає лише вузьку ділянку коду, не дьоргаючи решту системи. На рисунку 2.1 гарно показано загальну схему платформи LitExplorer та взаємодію між головними компонентами.

Платформа по суті спирається на три ядра. По-перше, модуль парсингу ParseNovelsFillDatabase: він на автопілоті тягне метадані з різних вебджерел, нормалізує теги й скидає все добро до бази. Запускається цей краулер за

розкладом або вручну, тож актуальність даних тримається без жодного дискомфорту для читача.

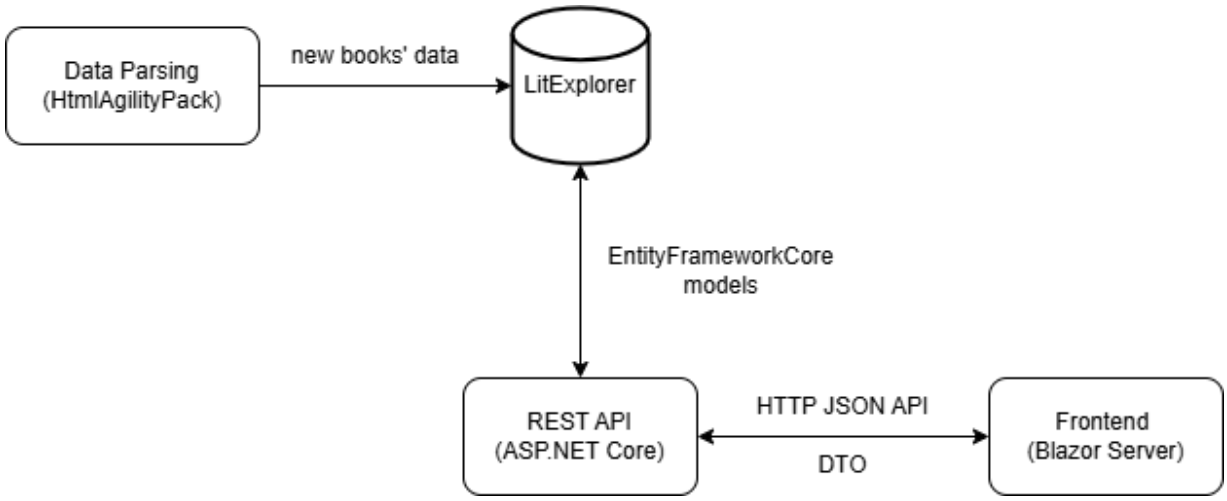

Рисунок 2.1 – Схема архітектури платформи обєднаного пошуку LitExplorer

По-друге, серверна частина – LitExplorerAPI на ASP.NET Core – віддає REST-інтерфейс до зібраного контенту. ASP.NET Core славиться «легким, високопродуктивним та модульним HTTP-пайплайном», підтримує Windows/macOS/Linux й обробляє пошук, фільтри та рекомендації спритно й стабільно.

По-третє, клієнтський сегмент реалізовано на Blazor Server: уся розмітка формується на сервері, а до браузера летять лише диф-оновлення через SignalR, що одночасно скорочує стартове завантаження й ховає C#-код від цікавих очей. Якщо раптом трафік пошуку зростає, достатньо докинути ресурсів саме під API, не чіпаючи ні краулерів, ні фронту – IBM окремо підкреслює подібну еластичність багаторівневих систем.

На користь обраної трирівневої архітектури варто згадати основні вигоди:

– незалежне масштабування. Кожен шар (парсинг, API, UI) нарощується окремо – хоч додавай воркерів для API, хоч клонуй краулери, лишаючи решту компонувань у спокої;

– легший дев і саппорт. Чітке розділення дозволяє різним командам паралельно крутити API, фронт або збирач даних, користуючись оптимальними для кожного інструментами;

– підвищена безпека. Фронт не сміє напряму лізти в базу: усі запити проходять крізь серверну логіку, яка править за внутрішній шлагбаум і майже нівелює SQL-ін’єкції та інші витівки.

Три головних модуля працюють через звичайні HTTP-запити й DTO-об’єкти. ParseNovelsFillDatabase з регулярним інтервалом доповнює базу даних свіжою інформацією. Коли клієнтський інтерфейс надсилає запит (пошук, фільтр, рекомендації тощо), серверний API тягне потрібне з бази, пропускає крізь бізнес-логіку (сортування, фільтри, формування підказок) і повертає результат у вигляді DTO. Далі ці відповіді перетворюються на UI компоненти, які Blazor Server малює у каталог літератури, хмари тегів, рекомендаційні блоки й форми авторизації.

Схема розподілу проєкту на шари дарує приємну гнучкість – кожен блок розв’язує свою проблему, тож налагоджувати й тестувати функціонал платформи значно легше. Також досягається масштабованість горизонтальним нарощуванням: додавання екземплярів API, посилення СУБД, тощо. Безпека також тримається на хорошому рівні в такій архітектурі – чутливі дані живуть тільки на сервері, а трафік між клієнтом та API проходить через захищений шар аутентифікації. Якщо з’явиться потреба у розширені можливостей застосунку, чітка модульність дозволить це зробити з максимальним комфортом для розробника.

Проаналізувавши всі перечисленні переваги можна зробити висновок, що описана структура вебплатформи прекрасно підходить системі в якої критично важливі продуктивність, масштабованість та надійна безпека.

2.3 Проєктування бази даних та схеми взаємодії з зовнішніми API

Головна реляційна база даних вебресурсу LitExplorer спроєктована з метою зберігання метаданих романів і профілів читачів. У серці схеми розташована таблиця Books, що містить унікальні записи творів у вигляді назви та числового ідентифікатора. Другою за значимістю є таблиця Authors, де знаходиться інформація про самих письменників.

Кросплатформена природа книжок – один і той саме роман може жити одразу на декількох майданчиках – показана у вигляді зв’язувальної таблиці BooksSources. Вона тримає зовнішні ключи на Books і Sources (джерела) та оригінальну URL адресу місця видання твору.

Усі метадані книги: автор, опис, середня оцінка читачів, кількість оцінок, глав, переглядів, і таке інше – розташовані в таблиці BooksMeta. Дана цифрова «анатомія» твору стосується сутності BooksSources, адже статистика та інформація про книгу може відрізнятися на різних вебресурсах.

Категоризація літератури реалізується парою сутностей TagsCategories та Tags. Перша описує самі рубрики (жанри, теми), а друга містить нормалізовані назви тегів. Відношення багато-до-багатьох між книгами та тегами представлено таблицею BooksTags, тож будь-яка книга може носити цілий набір тегів.

Користувацький сегмент бази даних також є упорядкованим. Таблиця Users зберігає облікові записи читачів, LibraryStatuses має набір можливих міток творів у персональній бібліотеці, Libraries тримає парний зв'язок користувач-книга, а ReadingHistory занотовує прогрес читання.

Дана архітектура формує зв’язну модель метаданих творів, авторів, тегів і користувачів, що лягає в основу роботи сервісу.

На рисунку 2.2 продемонстровано повну ERD-схему з усіма таблицями та їхніми відносинами. Ті ж структурні елементи деталізовано у таблиці 2.1, де можна швидко оглянути опис кожної реляційної сутності.

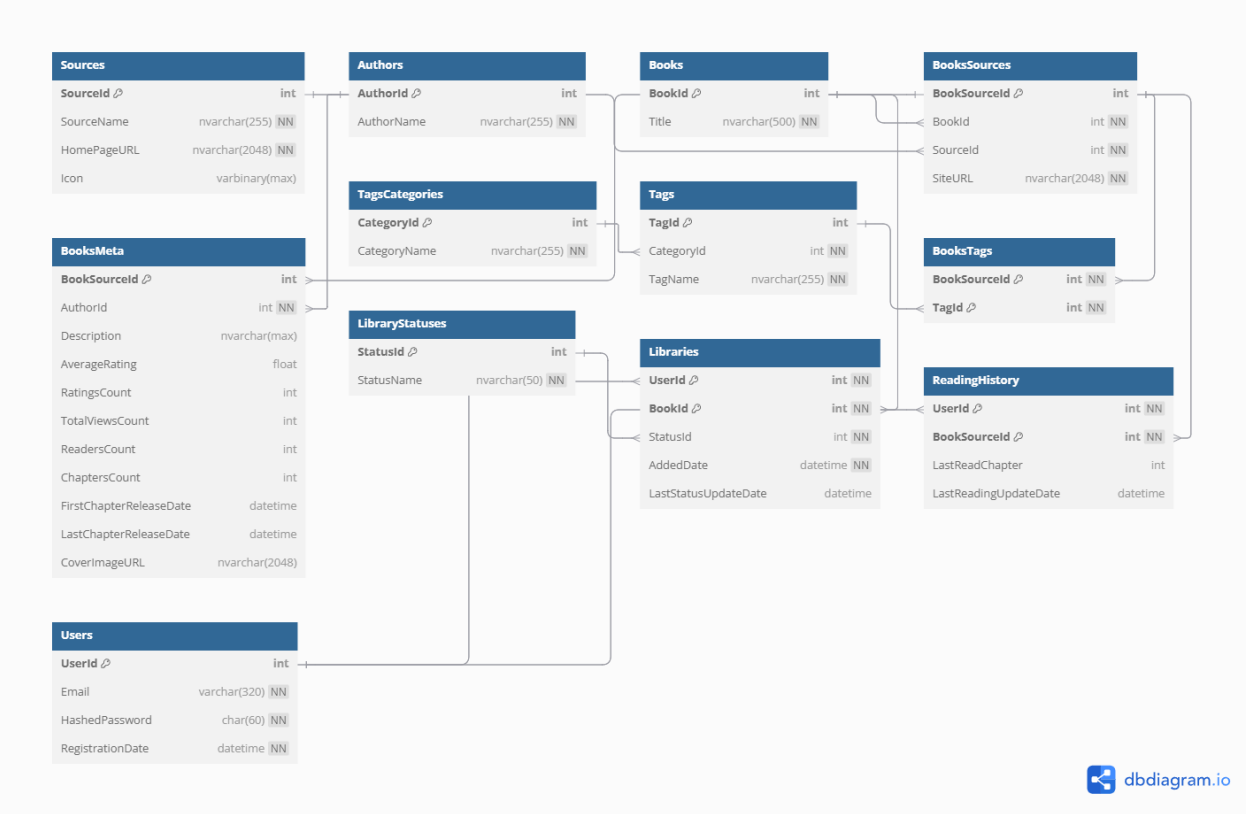

Рисунок 2.2 – Схема реляційних відносин бази даних LitExplorer

Таблиця 2.1 – Сутності основної БД

Сутність	**Опис**
Sources	Інформація про сайти-джерела романів і творів.
Authors	Дані про авторів романів (ім’я, біографія та ін.)
Books	Основні метадані творів (унікальні назва, ідентифікатор)
BooksSources	Зв’язки між творами та сайтами з додатковими метаданими (URL, рейтинг тощо)
BooksMeta	Детальні метадані творів (рейтинги, кількість переглядів, дата оновлення тощо)
TagsCategories	Категорії тегів (жанри, тематики)
Tags	Нормалізовані назви тегів
BooksTags	Зв’язок «багато-до-багатьох» між книгами та тегами
Users	Дані про зареєстрованих користувачів системи

Продовження таблиці 2.1

Сутність	Опис
LibraryStatuses	Можливі статуси книг у бібліотеці користувача («Читає», «Прочитано» тощо)
Libraries	Зв’язки між користувачами та книгами в їхніх особистих бібліотеках
ReadingHistory	Історія читання (прогрес) користувача по кожному твору

Разом із головною схемою існує відокремлена база нормалізації тегів. Вона гарантує, що різні назви з усіх можливих джерел зводяться до одного канону, тож пошук і рекомендації не плутаються у синонімах.

Основою цієї допоміжної БД служать знову-таки TagsCategories і Tags – перша задає самі рубрики, друга тримає канонічні назви. Щоб прив’язати локальні підписи сайтів до стандартного словника, для кожного джерела створено свою таблицю-словник: RoyalRoadsTags, ScribbleHubTags тощо. Кожен рядок містить альтернативне ім’я (AliasName) та зовнішній ID на канонічний тег. Така конструкція дає змогу будь-які, навіть екзотичні, позначки, що прилетіли з парсерів, негайно стикувати з єдиною внутрішньою назвою. Таблиця 2.2 наглядно перелічує усі сутності бази даних нормалізації тегів та їх опис.

Таблиця 2.2 – Сутності БД нормалізації тегів

Сутність	Опис
TagsCategories	Категорії тегів
Tags	Канонічні назви тегів (зі зв’язком із категорією)
RoyalRoadsTags	Відповідності альтернативних назв тегів сайту RoyalRoads
ScribbleHubTags	Відповідності альтернативних назв тегів сайту ScribbleHub

Далеко не кожен веброман-портал дає повноцінний API, тому LitExplorer покладається на фірмові парсери. Вони вичитують сторінки, витягують назву, автора, опис, теги, лічильники переглядів і все це добро одразу трансформують у внутрішній формат. Якщо в якоїсь платформи таки є обмежений API, то парсер бере, що дають, а решту дозбирає сам.

Отримана інформація спершу прямує або прямо в основну БД, або в таблиці нормалізації тегів – залежно від того, що саме витягнули. На виході маємо єдину, узгоджену структуру даних, тож рекомендації, пошук і особисті бібліотеки працюють без зайвих сюрпризів.

Frontend не повинен копатися в нутрощах бази даних, тому платформа спілкується з нею крізь звичайний REST-інтерфейс – клієнт надсилає HTTP-запити, а сервер відповідає JSON-пакетами. Між сирою схемою БД і зовнішнім контрактом стоїть окремий прошарок DTO (транспортних об’єктів даних). Для кожної таблиці в БД існує свій однойменний клас у просторі імен LitExplorerDTO. Наприклад, BookDTO тримає базову інформацію про твір і всередині одразу несе колекцію BookSourceDTO, кожен елемент якої містить свій BookMetaDTO та масив TagDTO. Подібно, TagsCategoryDTO повертає список категорій разом із вкладеними тегами.

Набір контролерів в API розділений за задачами. Один відповідає за перегляд і фільтрацію книжок: отримує структуру BrowseFilterDTO, виконує вибірку, а назад віддає пагінований перелік BookDTO. Інший виводить довідники тегів і категорій, ще один – список джерел. Окремі контролери обслуговують рекомендації та дії користувачів. Кожна кінцева точка приймає і вертає суворо фіксовані DTO-структури. Наприклад, запит BrowseBooks приймає параметр BrowseFilterDTO, БД дивиться відповідні записи, і сервер повертає масив об’єктів BookDTO з назвою, авторами, тегами та джерельною інформацією. Такий механізм роботи дає клієнтській частині готові типізовані дані і не прив’язує її до внутрішніх таблиць.

Також існують контролери метаданих, вони повертають допоміжну інформацію з бази даних, таку як набір можливих станів творів у

персональній бібліотеці (LibraryStatusDTO), оригінальних вебресурсів (SourceDTO), стандартизованих тегів та категорій (TagsCategoriesDTO з відносними TagsDTO).

Контролери відповідні за читача передають об’єкти UserDTO, LibraryDTO, ReadingHistoryDTO та інші у JSON-форматі. Завдяки такій уніфікованій схемі frontend отримує повний набір даних у зручному для роботи вигляді, а візуальні компоненти: каталоги творів, список тегів, джерел, персональна полиця, – малюються без зайвих перетворень.

2.4 Проєктування системи рекомендацій (AI-модуль та стандартні алгоритми)

2.4.1 Архітектура модуля рекомендацій і стандартні алгоритми

Модуль рекомендацій LitExplorer інтегрується до серверного API як окремий «AI-сервіс», що приймає запит клієнта і повертає список творів за різними критеріями. На рівні архітектури (рис. 2.3) цей модуль може працювати двома режимами: офлайн (побудова і оновлення моделей/кешу) та онлайн (формування результатів під час запиту).

Під час офлайн-етапу накопичуються історичні дані: інформація про книги, їхні теги, рейтинги, кількість переглядів і статистику за місяць, а також дії користувачів (наприклад, прочитані або вподобані книги). На їх основі обчислюються допоміжні дані – наприклад, будуються векторні уявлення книжок (TF-IDF опису, бінарні вектори категорій і тегів, нормалізовані метрики). Результати цього етапу зберігаються в кешованій таблиці (наприклад, BooksFeatures), щоб при онлайн-запитах не виконувати тяжких обчислень заново.

Після офлайн-підготовки, коли клієнтський фронтенд робить запит до /api/Recommendations, API передає на вході список опцій (BestOfMonth, Hottest, Personal тощо) і профіль користувача.

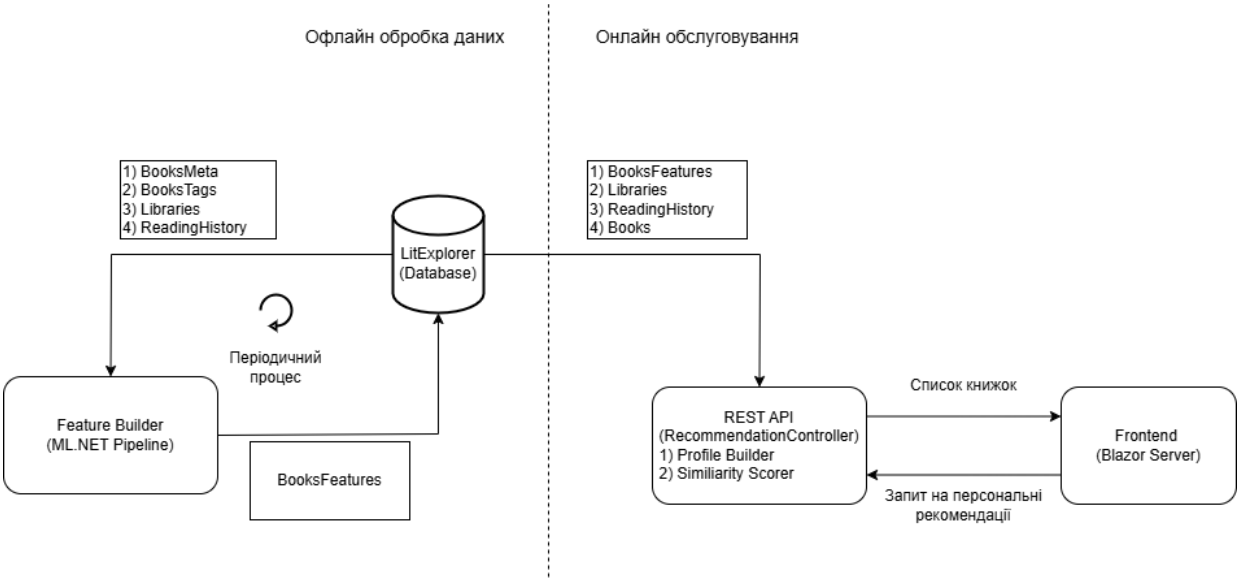

Рисунок 2.3 – Схема архітектури механізму персональних рекомендацій

Алгоритм відфільтровує або відсортує доступний каталог романів і повертає відфільтрований перелік. Наприклад, стандартні (неперсоналізовані) рекомендації формуються за такими «популярними» критеріями: Best of Month (топ за сумою рейтингів та активностей за останній місяць), Hottest (на базі динаміки переглядів або лайків), Новинки (найсвіжіші серіали). Ці фільтрації часто реалізуються як прості вибірки SQL або алгоритми сортування по метриках (рейтинг, дата, перегляди). Такий підхід дозволяє швидко показати загальні тренди незалежно від індивідуальних вподобань. Стандартні рекомендації корисні для збільшення охоплення: наприклад, показ «найпопулярнішого місяця» стимулює нових користувачів долучатися до обговорених творів, а «гарячих новинок» – утримувати інтерес.

Крім того, до стандартних алгоритмів можна віднести «вибірки по категоріям» (наприклад, найкращі за жанром чи тегом) або редакційні списки. Проте головною метою AI-модуля є саме персоналізовані рекомендації, адаптовані до читача. Структура формування користувацьких рекомендацій виглядає таким чином: історія читань і вподобань користувача формує профіль (онлайн-етап), який порівнюється з переліком інших книжок, відсортованих за релевантністю чи схожістю до профілю. Такий

двоетапний підхід («офлайн+онлайн») дозволяє швидко формувати персональні списки, оскільки важкі обчислення виконуються наперед.

2.4.2 Контентно-орієнтовані рекомендації

Контентно-орієнтовані алгоритми рекомендують книги, що схожі за змістом на ті, які вже сподобалися користувачу. У контексті LitExplorer «зміст» представлені наборами ознак книги: текстовий опис (або текст глав), жанрові та тематичні теги, інформація про автора, кількість розділів, рейтинг, метрики переглядів тощо. Передавання цих ознак у векторному вигляді дозволяє застосовувати міри схожості між книгами. Наприклад, короткий опис і назву книгу перетворюють у TF-IDF-вектори слів, теги – у one-hot-вектори, а числові характеристики (рейтинг, число розділів) нормалізують. У результаті кожній книзі відповідає «відбиток», тобто багатовимірний вектор Features.

При отриманні запиту профіль користувача можна зобразити як усереднений вектор із векторів книг, які він читав чи вподобав. Далі алгоритм обчислює косинусну чи іншу відстань між цим профілем і векторами всіх інших книг: найбільш близькі за кутом (коефіцієнт косинусної схожості ≈ 1) вважаються максимально релевантними. Переваги: контентні рекомендації добре працюють для нових книг (яких користувач ще не бачить) і зрозумілі з точки зору доменної логіки («підписник жанру *X* отримає ще книги того ж жанру»). Вони також не потребують даних інших користувачів (сусідній користувач, якого не було у системі, все одно отримає рекомендації за схожістю контенту). Недоліки: цей підхід тяжіє до «шийних коліс» – рекомендує переважно книги, схожі за уже вподобаними характеристиками, і рідше відкриває зовсім нові напрямки. Також створення і підтримка якісних векторних ознак (наприклад, правильно налаштованих TF-IDF-ваг чи структури категорій) потребує додаткової «доменної» роботи.

2.4.3 Порівняльний аналіз методів рекомендацій і обґрунтування вибору

Проєктуючи модуль персональних рекомендацій LitExplorer, доцільно розглянути три класичні підходи – контентно-орієнтований, колаборативний та гібридний варіант, у якому поєднано переваги перших двох. З погляду системного проєктування важливо з’ясувати, який алгоритм найкраще узгоджується з початковими даними платформи, очікуваними навантаженнями й еволюційною стратегією.

На першому етапі запуску LitExplorer у базі вже присутній багатий описовий контент (метадані книг, теги, категорії, розгорнутий текст анотацій), однак поведінкові дані про користувачів накопичуються поступово. Отже, контентно-орієнтований підхід (рис. 2.4) стає природною вихідною точкою: він не залежить від великої кількості оцінок і дозволяє відразу пропонувати релевантні твори, обчислюючи косинусну схожість між векторами «відбитків» книги й профілем читача. Такі «відбитки» формуються офлайн: TF-IDF для тексту, one-hot для тегів, нормалізовані числові метрики, у підсумку усе об’єднано в багатовимірний вектор Features. Онлайн-частина лишається легкою – досить усереднити вектори прочитаного й відсортувати решту книг за кутовою близькістю.

Колаборативна фільтрація (рис. 2.5), навпаки, демонструє повну силу лише тоді, коли в матриці «користувач × книга» з’являються тисячі й десятки тисяч взаємодій. З одного боку, вона відкриває нові жанри – можна рекомендувати твори, зовсім не схожі за тегами, але популярні серед «сусідів за смаком». З іншого боку, метод страждає від проблеми «холодного старту»: новий користувач не має історії, а нова книга ще не зібрала рейтингів. Для LitExplorer, котрий лише набирає перших користувачів, це критичний фактор.

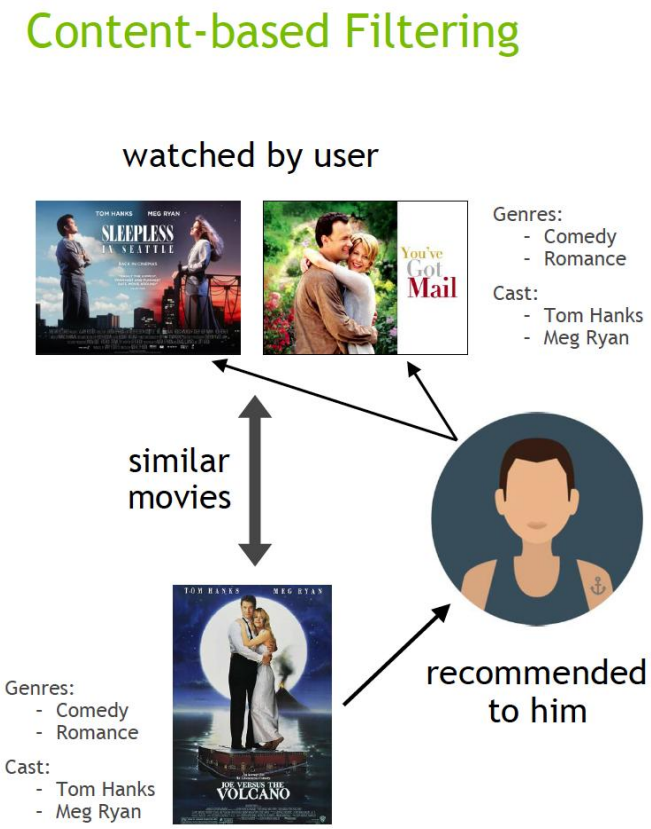

Рисунок 2.4 – Схема структури контентно-орієнтованого підходу [18]

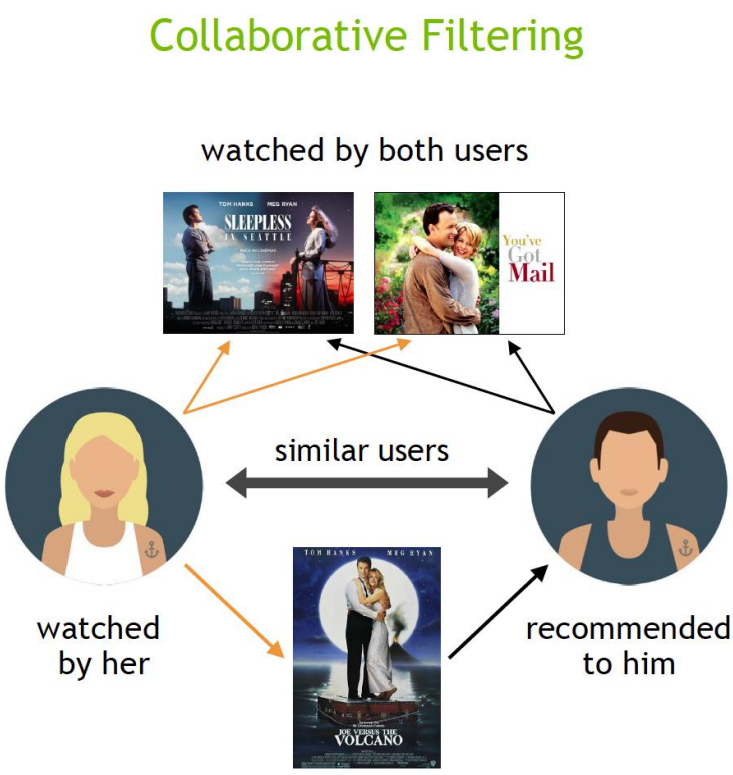

Рисунок 2.5 – Схема структури колаборційного типу рекомендацій [18]

Гібридний підхід розв’язує обидві крайнощі, поєднуючи результати із двох попередніх стратегій: контентний блок відповідає за глухі кути cold start, а колаборативний – за «ефект відкриття» та несподівані збіги. Але гібрид вимагає додаткової логіки злиття результатів (вагові коефіцієнти, навчання метамоделі) і складнішої інфраструктури моніторингу.

Ключові критерії добору алгоритму для першої ітерації LitExplorer:

– наявність багатого описового контенту вже на старті;

– невелика, але швидко зростаюча база користувачів;

– потреба пояснювати рекомендації (за тегами й жанрами);

– мінімізація ресурсоємних онлайн-обчислень;

– можливість безболісно розширити логіку, коли з’являться дані поведінки.

За сукупністю цих вимог прийнято рішення запустити персональні рекомендації на основі контентно-орієнтованої моделі з косинусною схожістю між вектором профілю користувача та кешованими «відбитками» книг. Водночас уже на рівні проєктування зарезервовано інтерфейси (окремі таблиці для матриці взаємодій і сервіс-процедури факторизації), аби з накопиченням статистики поступово зрушити до гібридної схеми – додавши другий канал колаборативних балів і зважене ранжування. Таким чином, система одразу забезпечує високу релевантність, а з часом – адаптивність і «серендипіті» без суттєвих змін API або бізнес-логіки.

У підсумку об’єднаний підхід «контент + резерв на колаборацію» задовольняє як початкові технічні обмеження, так і стратегічну потребу масштабувати рекомендації разом із ростом спільноти, зберігаючи при цьому прозорість і керованість модулю.

3 РЕАЛІЗАЦІЯ ПЛАТФОРМИ ОБ'ЄДНАНОГО ПОШУКУ АВТОРСЬКИХ ТЕКСТІВ

3.1 Обґрунтування вибору середовища програмної реалізації та додаткових інструментів розробки

Обраним програмним середовищем розробки платформи об'єднаного пошуку авторських текстів LitExplorer являється інтегральна частина екосистеми .NET – Visual Studio (рис. 3.1). Таке рішення дозволяє використовувати надійну мову програмування C# та набір підтримуваних бібліотек, які допоможуть з розробкою усіх шарів архітектури застосунку. Visual Studio має багато потужних інструментів, такі як редактор з підтримкою IntelliSense, процес відлагодження та аналізу коду, інтеграція з системами контролю версій.

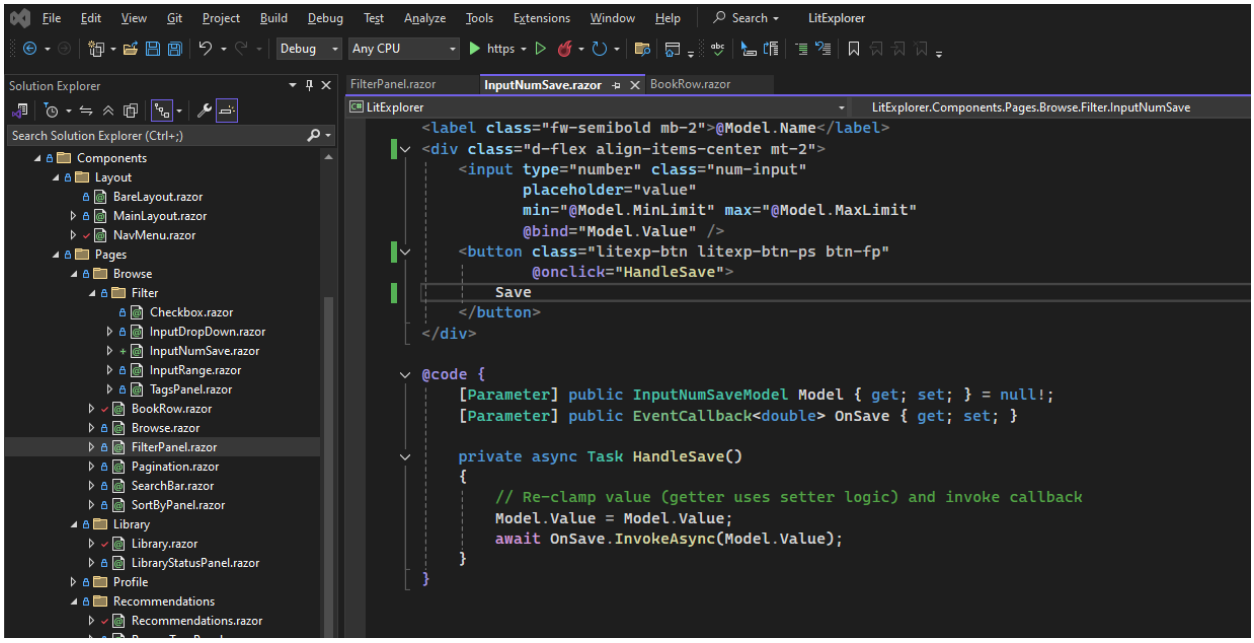

Рисунок 3.1 – Приклад інтерфейсу Visual Studio

Програмне середовище розробки Visual Studio також надає розробнику створити свої вебсервіси на основі варіативних шаблонів проєктів ASP.NET Core. Цей фреймворк є одним з основних ресурсів екосистеми .NET та

використовується у розробці серверної частини платформи LitExplorer через велику кількість переваг:

– висока продуктивність та масштабованість. ASP.NET Core демонструє рекордні результати у TechEmpower тестуваннях, що гарантує швидкий відгук API навіть під великою навантаженістю;

– кросплатформеність. Фреймворк можна розгорнути на Windows чи Linux без змін у коді, що забезпечує гнучкість розгортання сервісів у різних середовищах;

– чітка архітектура. Платформа LitExplorer побудована на шаблоні REST API, що використовує контролери з кінцевими точками. Такий підхід дозволяє роздробити роботу на декілька зон відповідальності і легше вносити нові зміни. Також механізм вбудованої ін’єкції залежностей спрощує структуру коду і полегшує юніт-тестування;

– потужна безпека. Технологія підтримує сучасні протоколи безпеки JWT, OAuth, автоматичне перенаправлення на HTTPS, захист від CSRF/XSS та інші механізми шифрування;

– інтеграція з базою даних. Доступ до баз даних в проєктах ASP.NET Core частіше реалізовано за допомогою офіційного ORM для .NET, бібліотеки EntityFrameworkCore. EF Core дозволяє працювати з даними через LINQ-запити та автоматичні міграції схеми, що спрощує проєктування запитів фільтрації за авторами, тегами, рейтингом тощо. Завдяки цьому обробка складних запитів пошуку в каталозі книг виконується ефективно;

– вбудована документація API. ASP.NET Core надає нативну підтримку Swagger через пакети Swashbuckle. Swagger автоматично генерує інтерактивну документацію REST-інтерфейсу LitExplorer та візуально демонструє в окремій вебсторінки браузеру, що полегшує тестування та подальший розвиток API.

Для реалізації модуля персональних рекомендацій була застосована кросплатформена технологія ML.NET. Фреймворк дозволяє створювати і тренувати моделі безпосередньо в середовищі .NET, використовуючи вже

відомі розробнику інструменти і бібліотеки. Така перевага дає можливість легко інтегрувати нейроні алгоритми рекомендацій (наприклад, матричну факторизацію) в існуючий код платформи, не змінюючи середовище розробки.

ML.NET підтримує такі операційні системи як Windows, Linux та macOS, забезпечуючи портативність модуля рекомендацій при розгортанні його на будь-якому сервері.

Ще однією сильною стороною фреймворку машиного навчання є підтримка сучасних алгоритмів рекомендацій, зокрема матрична факторизація. Також дозволяє підключати моделі з TensorFlow, ONNX та інших платформ, гарантуючи можливість розширення модуля рекомендацій новими підходами.

Для реалізації клієнтської частини було застосовано фреймворк Blazor, а саме шаблон проєкту Blazor WebApp. Технологія Microsoft інтегрує стандартний процес розробки вебсайту в екосистему .NET, задля побудови застосунку з використанням багатьох підтримуваних інструментів та бібліотек. Подібний підхід дає можливість використовувати мову програмування C# прямо в коді HTML, додаючи логіку, яка оброблюється на серверній частині застосунку.

Методом візуалізації було становлено значення InteractiveServer, що спочатку будує DOM вебсторінки на сервері і тільки потім відправляє готовий результат клієнту. Це дозволяє повністю розділити проєкт на зони клієнтської відповідальності та логіки взаємодії з дистанційним API.

Обраною системою контролю версій являється старий та досі популярний Git, що відомий своєю швидкістю та гнучкістю. Кожен проєкт що складає вебплатформу об’єднаного пошуку авторських текстів має велику кількість своїх власних гілок (рис. 3.2). Інструмент дозволяє легко об’єднувати зміни, відстежувати історію проєкта та перестрибувати на потрібну версію застосунку. Інтеграція Git у Visual Studio забезпечує зручний командний робочий процес. Дана технологія присутня у більшості сучасних

проєктів та являється системою з відкритим кодом, що гарантує широку та тривалу підтримку спільноти.

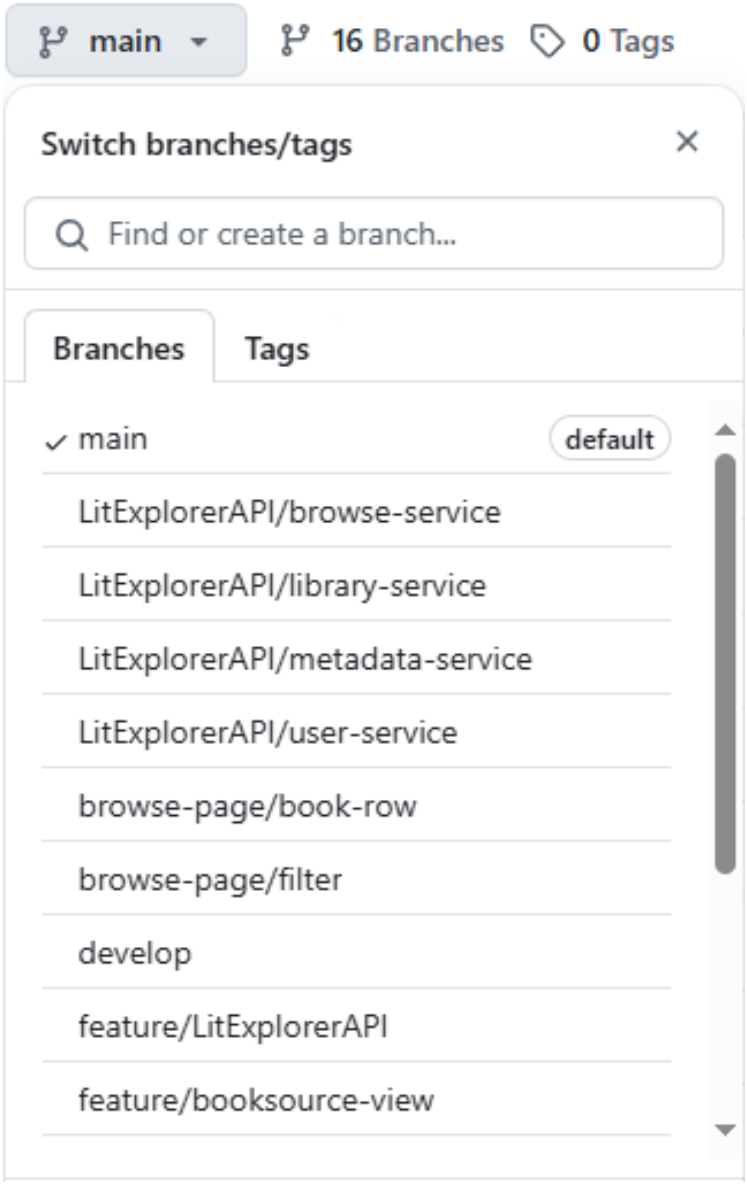

Рисунок 3.2 – Приклад гілок розробки серверної частини платформи

Отже, обрані середовище розробки та допоміжні інструменти повністю задовольняють усім можливим потребам реалізації платформи об’єднаного пошуку авторських текстів. Вони гарантують продуктивну розробку, високу швидкість обробки запитів, гнучку роботу з даними та зручність документування і підтримки сервісу, що робить їх гарним вибором для даної кваліфікаційної роботи.

3.2 Розробка застосунку отримання та зберігання даних книг з оригінальних джерел

3.2.1 Створення механізму пошуку та зберігання головної інформації книг

Проєкт ParseNovelsFillDatabase у межах екосистеми LitExplorer виконує завдання первинного виявлення літературних новинок на зовнішніх ресурсах і наповнення базових таблиць Books та BooksSources. Мета цього модуля – з мінімальними витратами часу охопити якомога більшу частину каталогів вебплатформ (передусім Royal Road, а за необхідності Scribble Hub), зафіксувати появу нових позицій і занести до БД абсолютний «скелет» кожного твору: його назву, адресу сторінки та джерело публікації. Подальші глибші атрибути, такі як описи, теги чи рейтинги, обробляються іншими підсистемами; тут же зосереджено виключно «розвідку» списків.

Потік роботи починається з отримання HTML-сторінки каталогу (рис. 3.3), що містить перелік романів певної категорії, жанру або рейтингу. У структурі документа модуль визначає окремі картки творів, із кожної виокремлює текст заголовка і канонічне посилання на сторінку роману. Водночас фіксується назва самого сайту-джерела, що пізніше трансформується у запис таблиці Sources. У разі, якщо книга з таким заголовком уже присутня в Books, її ідентифікатор просто повторно використовують; якщо ні – створюють новий рядок і повертають згенерований BookId. Далі формується пара BookId–SourceId і разом з URL сторінки записується у BooksSources, уникаючи дублювань завдяки унікальному обмеженню на цю пару. Таким чином база росте горизонтально: кожен твір з’являється лише раз, а всі його представлення на різних сайтах пов’язуються відокремленими записами BooksSources.

У консольному вікні програми (рис. 3.4) кожний крок відображається стисло: зазначається поточний URL, кількість виявлених карток та статистика збережених рядків. Такі повідомлення дозволяють

відслідковувати прогрес і, при необхідності, повернутись до проблемної сторінки.

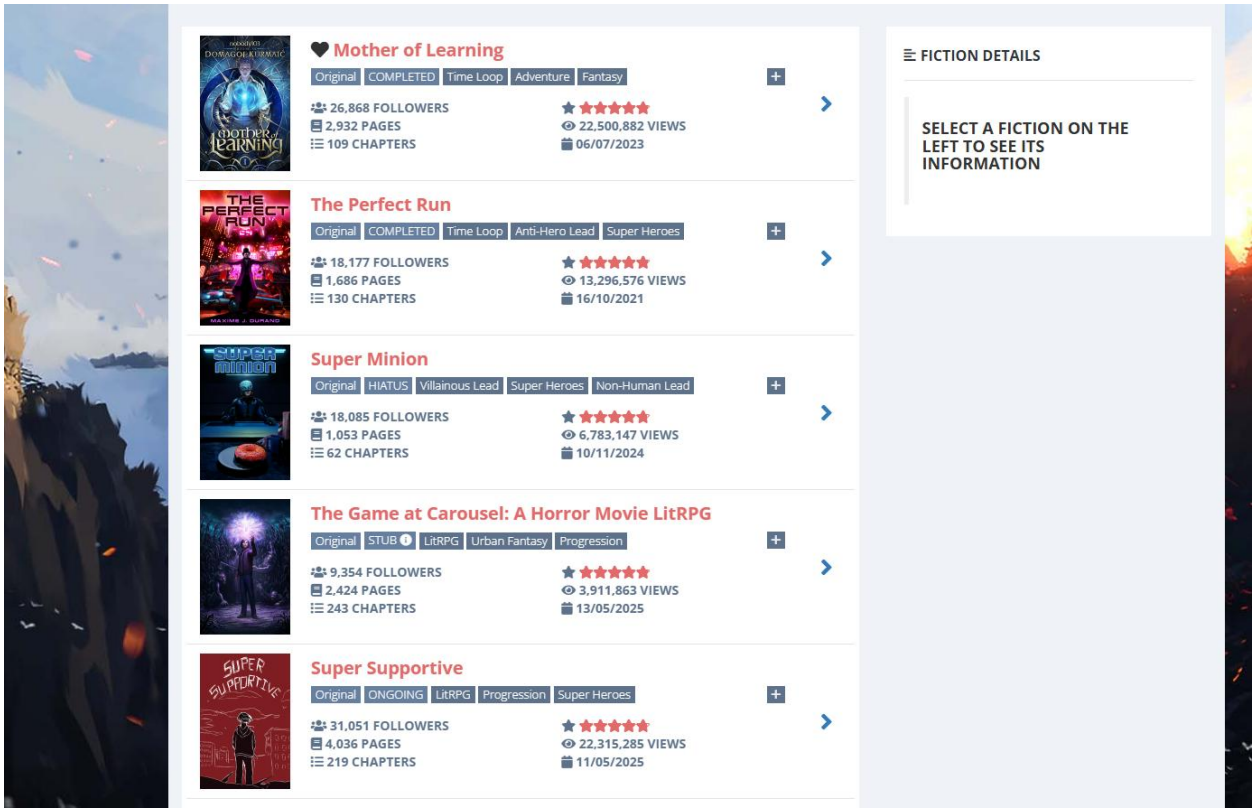

Рисунок 3.3 – Приклад сторінки каталогу із списком романів

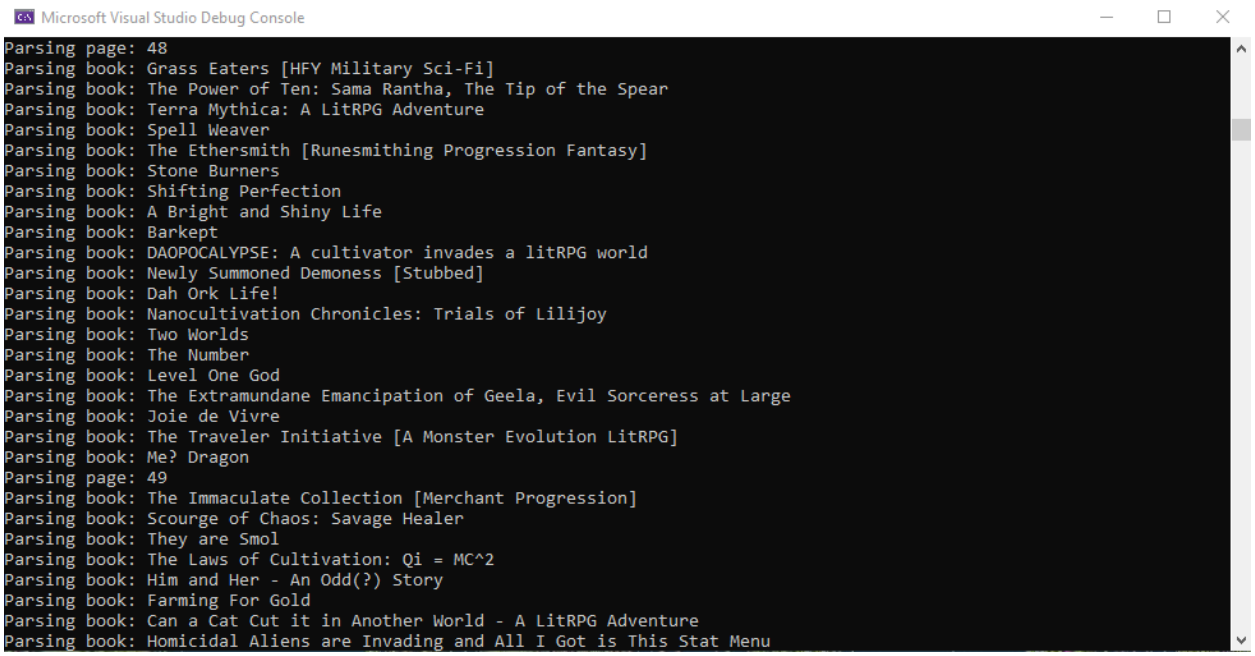

Рисунок 3.4 – Вивід консолі застосунку під час сканування каталогу

Завдяки поділу на короткі запити та відмові від глибокого аналізу контенту модуль працює максимально швидко, що важливо для щоденної синхронізації великого обсягу сторінок. Він не блокує ресурси системи та не потребує складних механізмів контролю повторів: достатньо перевірити, чи існує заголовок у Books і чи зафіксовано зв’язку в BooksSources. Отже, зазначений механізм формує першу ланку повного конвеєра LitExplorer, забезпечуючи своєчасне виявлення нових творів і закладаючи фундамент для наступних етапів збагачення метаданими та рекомендаціями.

3.2.2 Створення механізму пошуку та зберігання головної інформації книг

На другому етапі обробки даних платформи LitExplorer відбувається збір детальних метаданих про виявлені романи. Отримані раніше записи з бази даних про знайдені книги (їхні назви та URL-адреси) використовуються для поетапного завантаження повних сторінок творів з вебресурсів. Кожен такий вебдокумент містить набір інформації про книгу, який необхідно витягти: це автор твору, його короткий опис чи анотація, середній рейтинг і кількість оцінок, статистика переглядів чи читачів, число розділів, дати публікації (першого й останнього розділу), зображення обкладинки, а також набори тегів, що характеризують жанр і тематику. Модуль вилучення метаданих обробляє HTML-код цих сторінок, знаходячи відповідні елементи (заголовки, блоки тексту, графічні об’єкти тощо) і перетворює їх у структуровані дані. Зміст кожної сторінки обробляється послідовно: спочатку витягуються заголовок і автор, потім збираються числові показники (рейтинг, перегляди, кількість розділів) та текстові поля (опис, теги). За необхідності здійснюється очистка й форматування тексту (наприклад, відділення значень числа переглядів від підпису «Views» тощо).

На рисунку 3.5 показано типову сторінку роману з одного з онлайн-ресурсів, яку обробляє система. Вона містить обкладинку твору, назву, ім’я автора, опис, рейтингові зірочки, цифри переглядів і списки тегів. У процесі роботи модуля такі елементи сторінки автоматично «зчитуються» і перетворюються в набір параметрів. Наприклад, текст опису книги з вебсторінки зберігається у відповідному полі, ім’я автора порівнюється з існуючими в базі (або додається як новий запис), а теги зчитуються як окремі рядки. За рахунок узагальнення та структурування даних на виході формується однорідний набір метаданих для кожного роману незалежно від формату вихідної вебсторінки.

Рисунок 3.5 – Сторінка книги на вебресурсі, з якої витягуються метадані

Після вилучення необхідної інформації модуль виконує збереження отриманих метаданих у реляційній базі даних LitExplorer. Опис книги, рейтинг, кількість переглядів та інші числові характеристики записуються в таблицю BooksMeta. Інформація про авторів ведеться окремо: ім’я автора

зберігається (або оновлюється) у таблиці Authors, а зв’язок між книгою і автором вказується через ідентифікатори записів. Це забезпечує єдину базу даних авторів, до якої можуть посилатися кілька творів одного автора. Теги кожної книги перед збереженням перетворюються на однаковий формат (ідентифікатор тегу) і записуються через проміжну таблицю BooksTags, що пов’язує книгу (запис у таблиці BooksSource або BooksMeta) з наборами усіх її тегів. Таким чином, кожна витягнута характеристика роману потрапляє у відповідну таблицю бази: текст опису і числові показники – у BooksMeta, автор – в Authors, теги – у BooksTags. Ця організація даних дозволяє уникнути дублювання інформації (наприклад, повторних введень однакових авторів) та забезпечує зв’язність даних у межах єдиної схеми.

Окрім запису метаданих у базу, під час виконання процесу модуль виводить у консоль хід операції для зручності контролю та налагодження.

На рисунку 3.6 представлено типовий приклад логів консолі в процесі вилучення даних. Тут можна побачити інформацію про поточні дії – наприклад, номер опрацьованої книги, статус збереження полів, попередження чи помилки при відсутніх даних тощо. Такий вивід допомагає розробникам відстежувати успішність отримання та запису кожної частини метаданих безпосередньо в процесі роботи модуля.

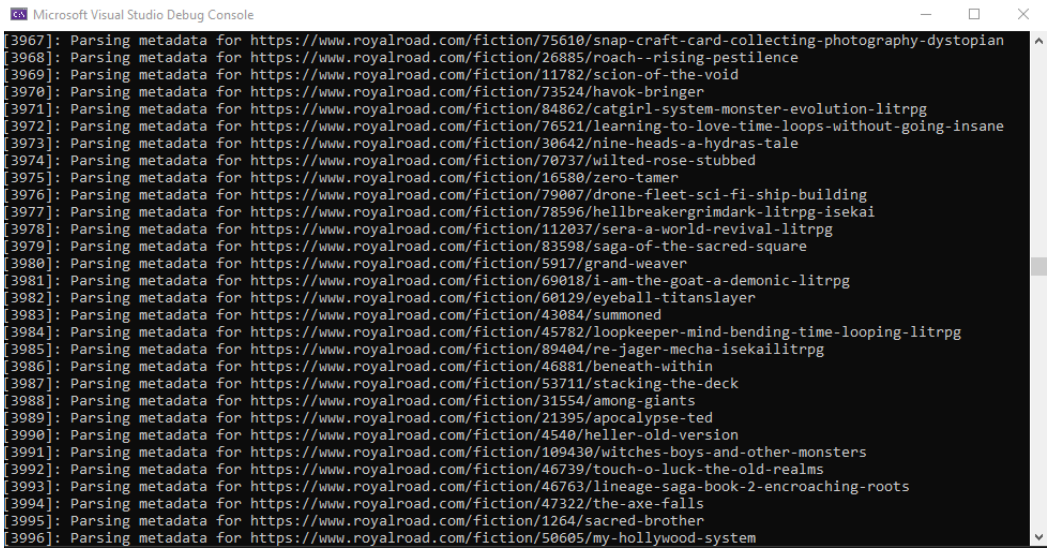

Рисунок 3.6 – Вивід консолі під час обробки метаданих книги

Особлива увага приділяється уніфікації тегів, які можуть надходити з різних сайтів у довільному вигляді. Адже один і той самий жанр або тема на різних вебплатформах може мати різні назви або синоніми. Щоб поєднати їх до єдиного формату, система звертається до окремої допоміжної бази TagDatabase (зберігається у Tag Normalization Database). У цій базі містяться таблиці відповідностей, де конкретні «сирі» назви тегів з кожного ресурсу зіставлені з універсальними категоріями та ідентифікаторами тегів платформи. При обробці даних модуль виконує пошук відповідностей у таблицях-посередниках – отримані з вебсторінки теги перетворюються на значення зі спільного словника платформи. Таким чином, після збереження процесу нормалізації всі книги в базі мають теги, приведені до єдиної таксономії LitExplorer, що дозволяє послідовно фільтрувати й аналізувати романи незалежно від їхнього оригінального джерела.

Таким чином, створений модуль отримання метаданих виконує ключову функцію другого етапу конвеєра збору даних системи LitExplorer. Він перетворює необроблені вебсторінки в структуровані записи в базі даних, формуючи повний профіль кожної книги. Це забезпечує необхідну інформаційну основу для наступних кроків – від запитів REST-API до побудови рекомендованих списків – і гарантує, що кожен роман із різних джерел зберігається в системі з уніфікованими метаданими. У результаті платформа отримує узгоджений каталог романів із єдиною схемою даних і таксономією тегів, готовий до подальшої обробки та подачі користувачам.

3.3 Розробка та налаштування серверної частини (backend, REST API)

Серверна частина платформи LitExplorer являється вебсервісом ASP.NET Core на основі технології REST API. Проєкт відповідає за взаємодію між головною базою даних та клієнтським застосунком, приймаючи запити від клієнта і повертаючи потрібні дані.

Технологія REST обробляє запити за допомогою контролерів. Це класи із спеціалізованим набором методів, кожен з яких представляє собою кінцеву точку API. При надходженні запиту відповідний контролер читає або змінює дані в базі через EntityFrameworkCore і повертає відповідь у форматі JSON.

Створення окремого застосунку для роботи з базою даних є гарним підходом для сегрегації відповідальності між частинами платформи, робить розробку легше та надає можливість масштабування і розширення проєкту.

Дані для обміну між клієнтами та вебсервісом представлено у вигляді DTO-структур (Data Transfer Objects). DTO відокремлюють внутрішню модель бази даних та надають лише ту інформацію, яку буде зручно передавати клієнту. Кожен об’єкт бази даних було спрощено, аби залишилися лише необхідні поля. Такий підхід дозволяє приховати зайві властивості і змінювати форму відповіді, не змінюючи структуру бази даних.

Процес тестування та перевірки роботи всіх методів REST API проходить через вбудований Swagger UI (рис. 3.7). Цей інструмент автоматично генерує інтерактивну вебсторінку з документацією API, де перераховані всі доступні кінцеві точки. Кожен метод розширюється для перегляду параметрів і опису, а також може бути запущений у браузері. Технологія облегшує роботу розробникам завдяки усуненню необхідності написання окремих додаткових сервісів тестування.

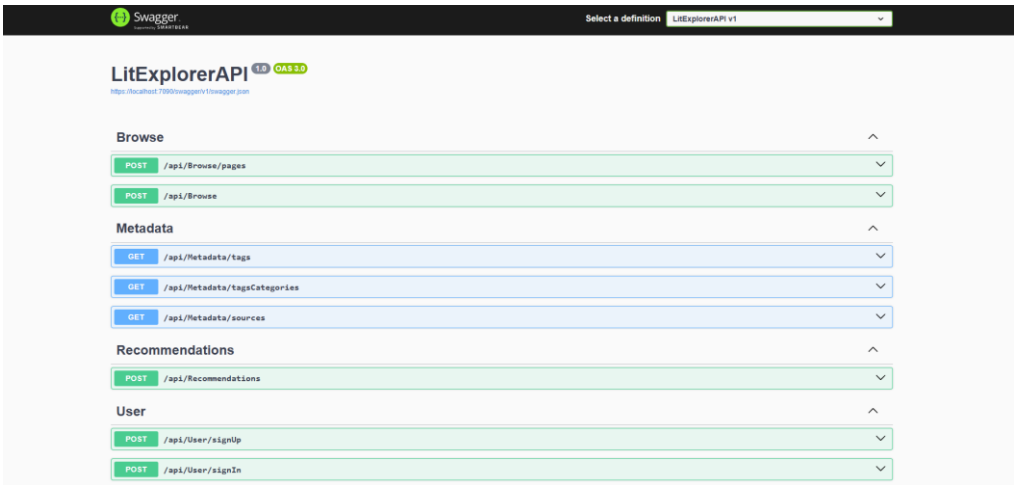

Рисунок 3.7 – Інтерфейс вебсторінки Swagger з кінцевими точками API

Підбиваючи підсумки, архітектура серверної частини LitExplorer чітко розділена на контролери за предметною областю і використовує відокремлені DTO-об’єкти для обміну даними. Клієнтський застосунок отримує зручний набір об’єктів JSON через стандартизоване REST API, а розробники мають наочний Swagger UI для перегляду та тестування всіх методів. Така структура платформи гарантує чіткий обмін інформацією між frontend і backend, відповідаючи принципам побудови сучасних вебсервісів.

3.4 Програмна реалізація клієнтської частини (Blazor WebApp)

3.4.1 Сторінка рекомендацій

Домашня вебсторінка за замовчуванням, що буде завантажуватися для усіх користувачів інтернету при переході на вебресурс LitExplorer – сторінка рекомендацій. Ця частина клієнтського інтерфейсу ставить за мету надати клієнту збірку з максимум 12 творів за обраною категорією:

– найкращі книги в поточному місяці;
– нові твори поточного місяця;
– персональні рекомендації.

Компоновка сторінки дуже проста. Вона містить в собі назву типу обраної рекомендації, кнопку з панеллю вибора категорії (рис. 3.8) та вертикальний список книг.

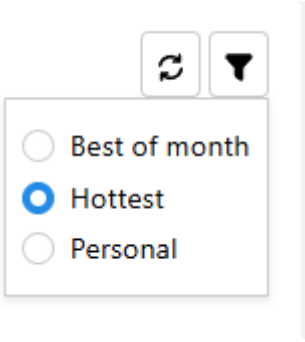

Рисунок 3.8 – Компонент вибору категорії рекомендації

Розглянемо візуально повну структури даної вебсторінки за допомогою її стандартного ви (рис. 3.9) – рекомендації найкращих книг в поточному місяці.

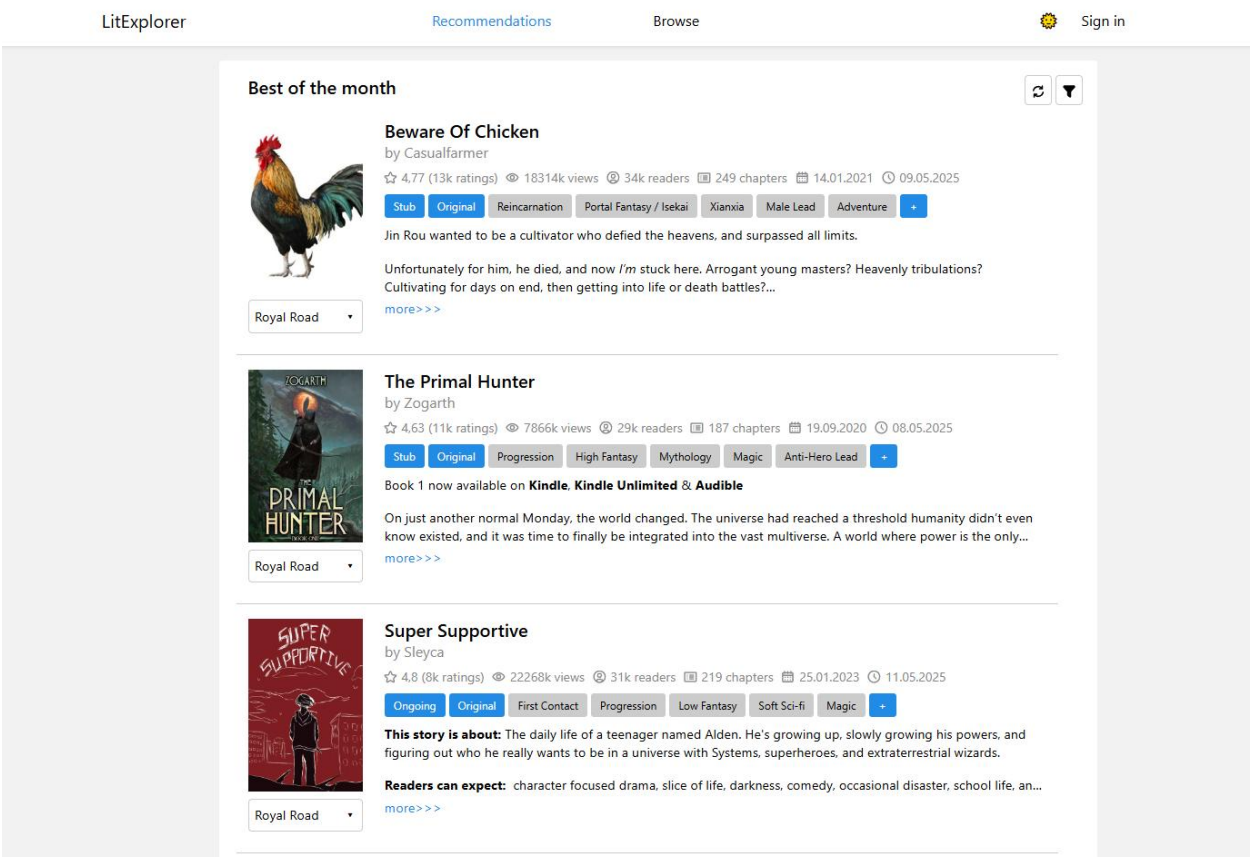

Рисунок 3.9 – Сторінка рекомендацій за категорією «Best of month»

Результат створеного списку обчислюється на основі інформації кількості та якості читацьких оцінок за поточний місяць. Елементи сформованого листа виведено за спаданням вказаних статистичних даних.

Категорія «Hottest» спрямована на просування нових авторських текстів та привертання уваги потенційних читачів до них. Наповнюючий контент списку формується лише за однім критерієм – дата публікації першої глави повинна бути в поточному місяці. На рисунку 3.10 можна побачити результати рекомендації нових книг за травень 2025.

Останній тип рекомендацій – персональні. Він базуються на літературній історії користувача вебплатформи та поточному наповненню

персональній бібліотеці. На відміну від перших двох опцій рекомендацій, третій видає кожен раз новий результат при перезавантаженні сторінки (рис. 3.11).

Персональний тип підтримує холодний підхід – коли користувач не авторизований в системі. В такому випадку результати роботи алгоритму будуть випадковими.

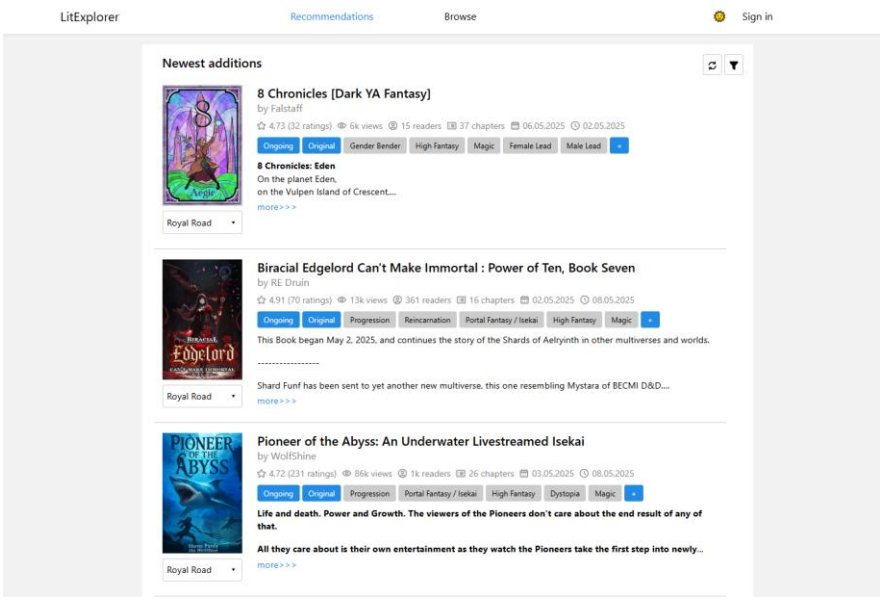

Рисунок 3.10 – Результат категорії рекомендації «Hottest»

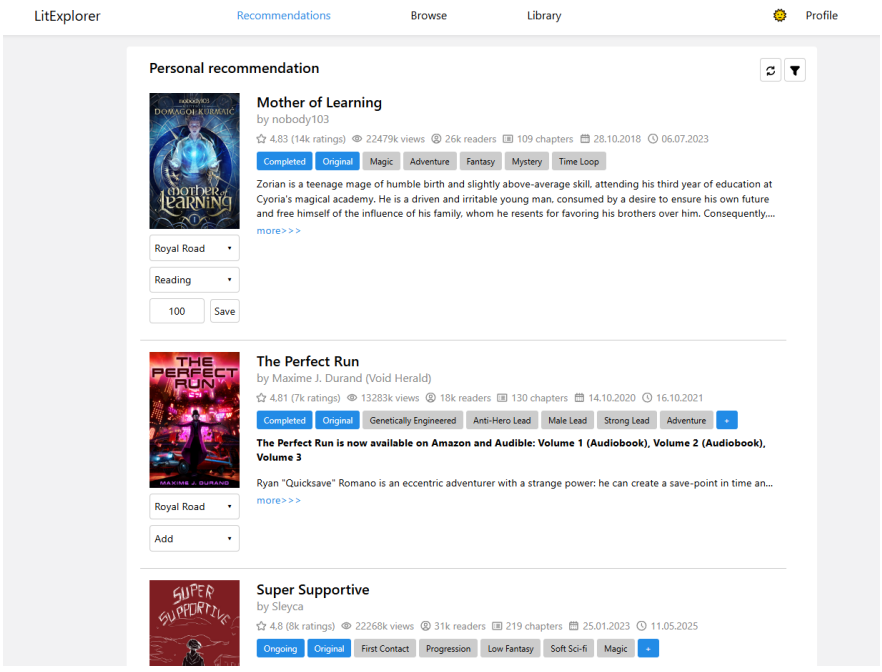

Рисунок 3.11 – Персональні рекомендації

3.4.2 Сторінка каталогу

Другою головною вебсторінкою клієнтського інтерфейсу платформи LitExplorer є сторінка розширеного пошуку (рис. 3.12). Функціонал даної частини вебсайта не залежить від стану авторизації користувача.

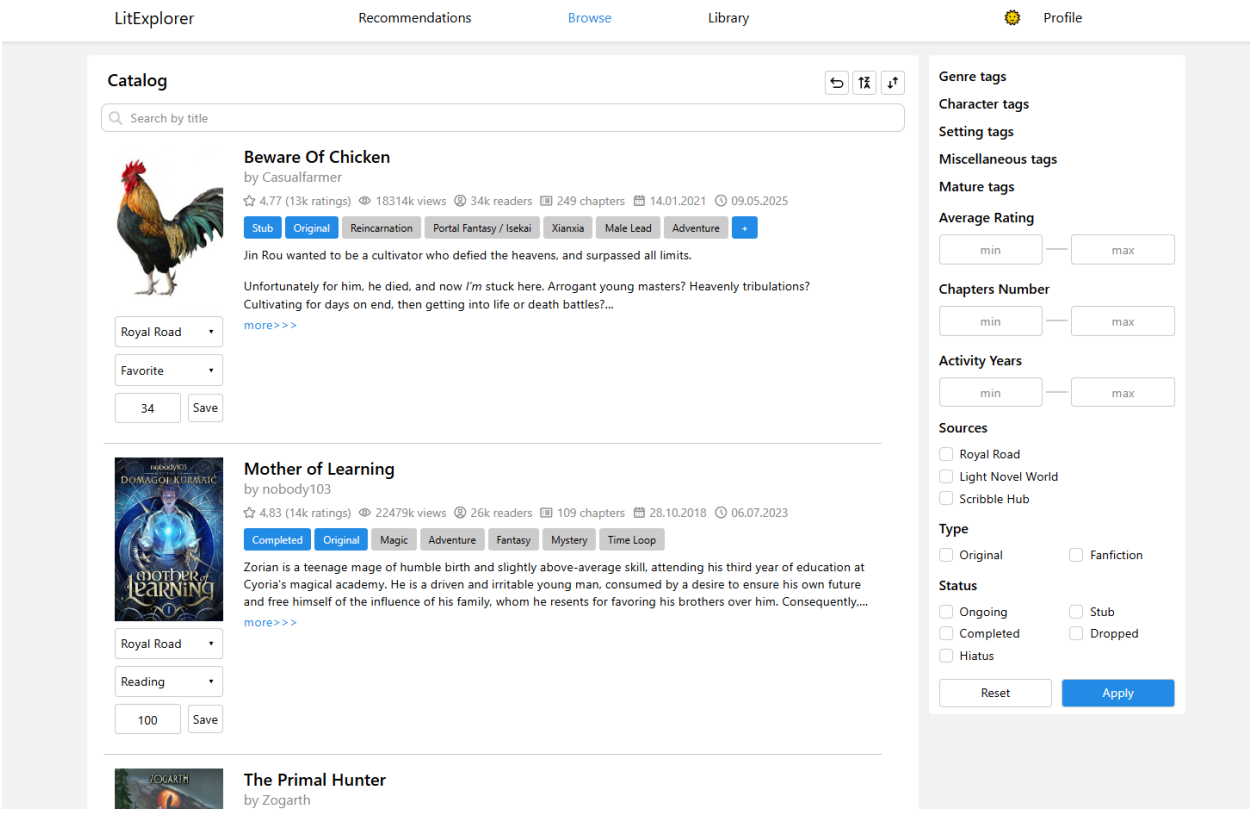

Рисунок 3.12 – Сторінка каталогу «Browse»

Структурно вебсторінка розділена на дві частини: список книг, які підходять за встановленими критеріями, та компонент фільтрації у вигляді панелі.

Перша колонка також має компонент пошуку літератури за вказаною назвою. Операція потребує точності в вказуванні інформації, або можна зробити пошук за ключовим словом. На рисунку 3.13 зображені книги, які задовольняють умову схожості з введеним ключовим словом.

Також існують дві панелі вибору типу сортування контенту каталогу: за метаданими (рис. 3.14), за порядком відображення (рис. 3.15).

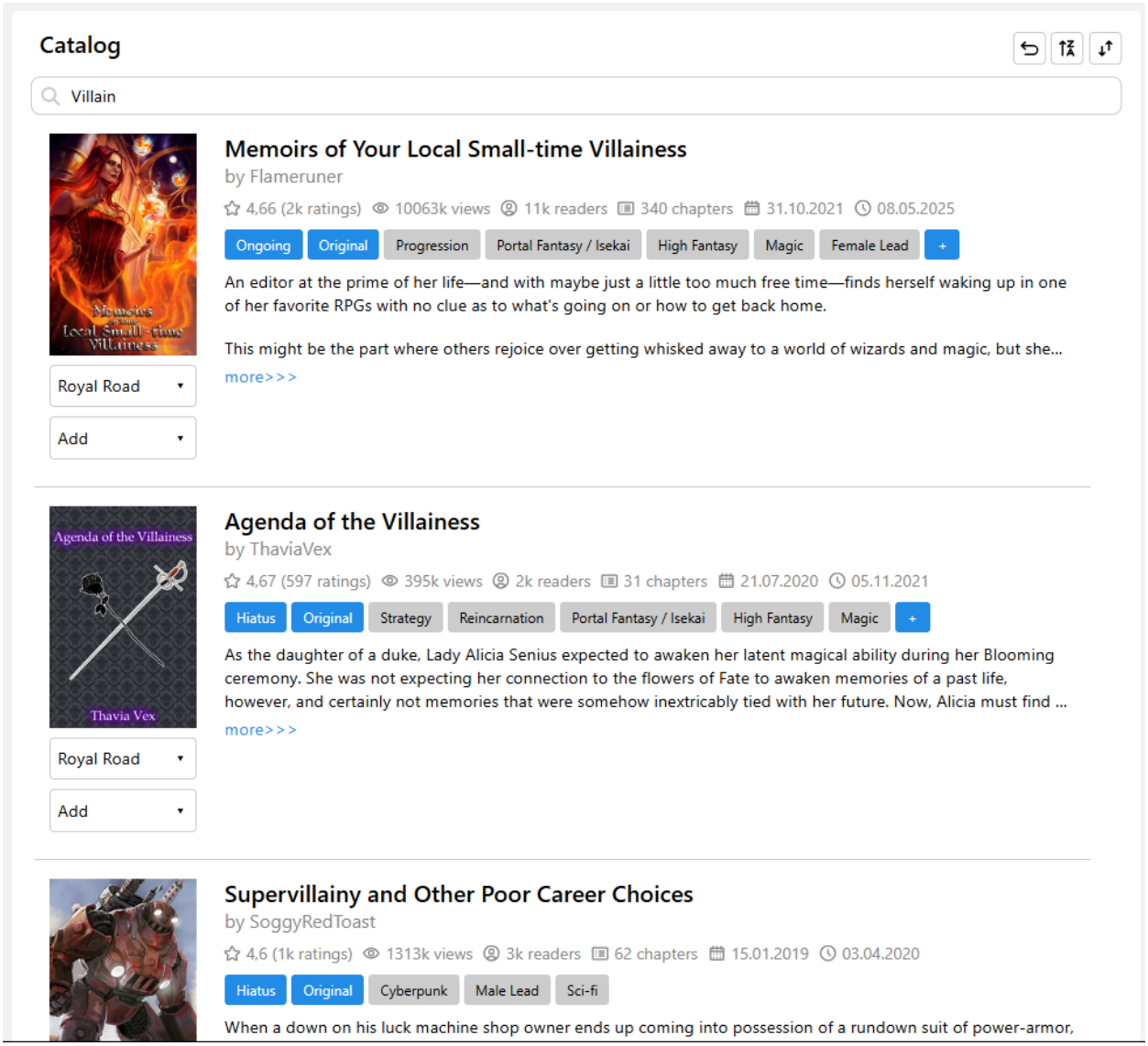

Рисунок 3.13 – Результат пошуку за назвою

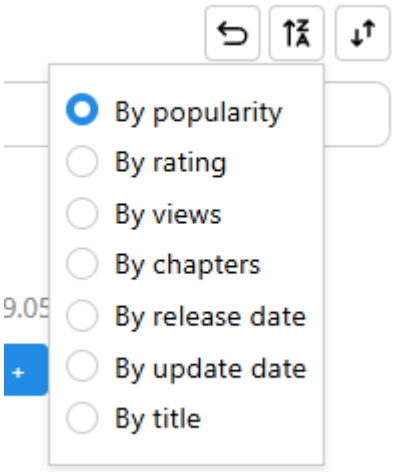

Рисунок 3.14 – Панель сортування книжок за метаданими

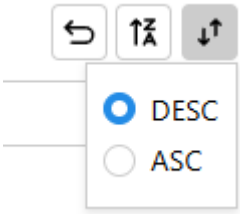

Рисунок 3.15 – Панель сортування книжок за спаданням або за зростанням

На відміну від вебсторінки рекомендацій, кількість авторських текстів наданих користувачу частіше забагато більше ніж 12. Подібне штучне правило не діє на сторінці каталогу в повній мірі. Вебсторінка «Browse» все ще має обмеження на надання максимум лише 12 творів, проте каталог має можливість продемонструвати усі книги, що є в базі даних, які задовольняють обраним опціям фільтрації та сортування. Для перегляду усіх доступних варіантів необхідно лише скористуватися компонентом навігації (рис. 3.16) за списком, який знаходиться у самій нижній частині вебсторінки, прямо під списком літератури.

Рисунок 3.16 – Компонент навігації за результатами розширеного пошуку

Кнопка «First» перенесе на першу дванадцятку списку, а «Last» на останню. Кнопки «Previous» та «Next» переміщуються на попередню, або наступну п’ятірку сторінок. Між цими опціями навігації можуть бути максимум лише 5 індексів сторінок.

Розглянемо тепер другу, праву частину структури вебсторінки каталогу. Майже усе вільне місце займає один компонент - панель фільтрації. На рисунку 3.17 можна гарно побачити які саме елементи приймають участь у процесі відбору книжок. Більшість місця займають теги, а остаток – деякі метадані: діапазони читацьких оцінок, кількості глав та роки активності автора.

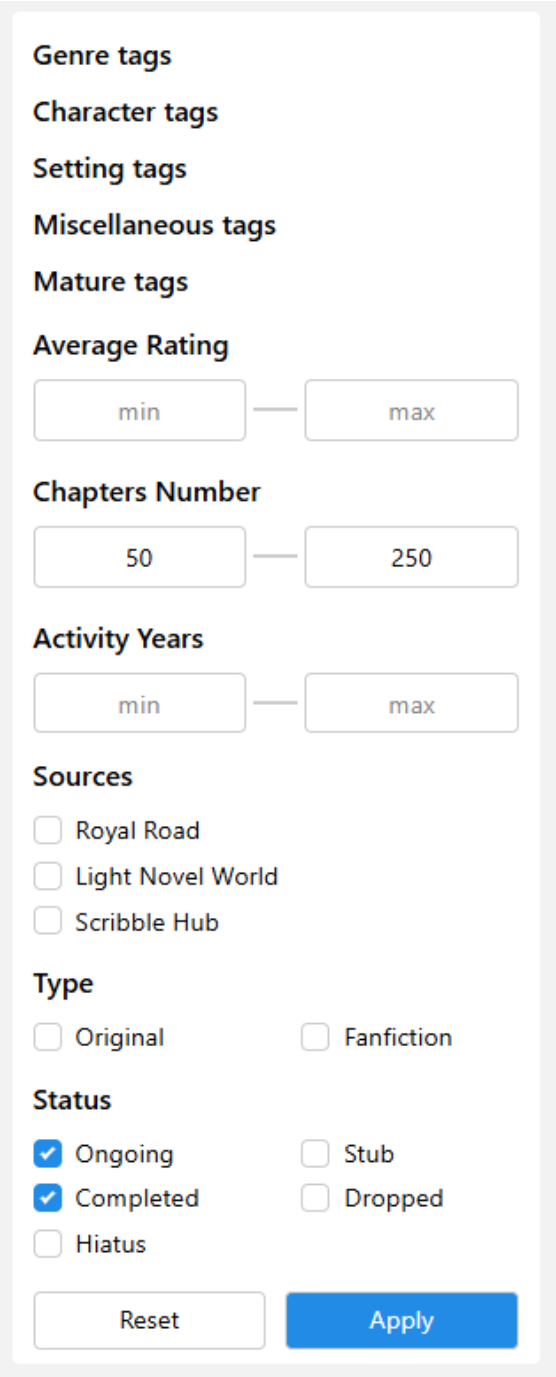

Рисунок 3.17 – Компонент панелі для фільтрації текстів

Усі checkbox UI-елементи, окрім розділу Sources, являються тегами різних категорій. Усього типів тегів 7, та вони працюють за принципом логічного «АБО».

Через завелику кількість тегів, більшість були розміщені в свої невеличкі вкладки панелі (рис. 3.18), які з’являються зверху основної, коли клієнт нажимає на перші п’ять категорій.

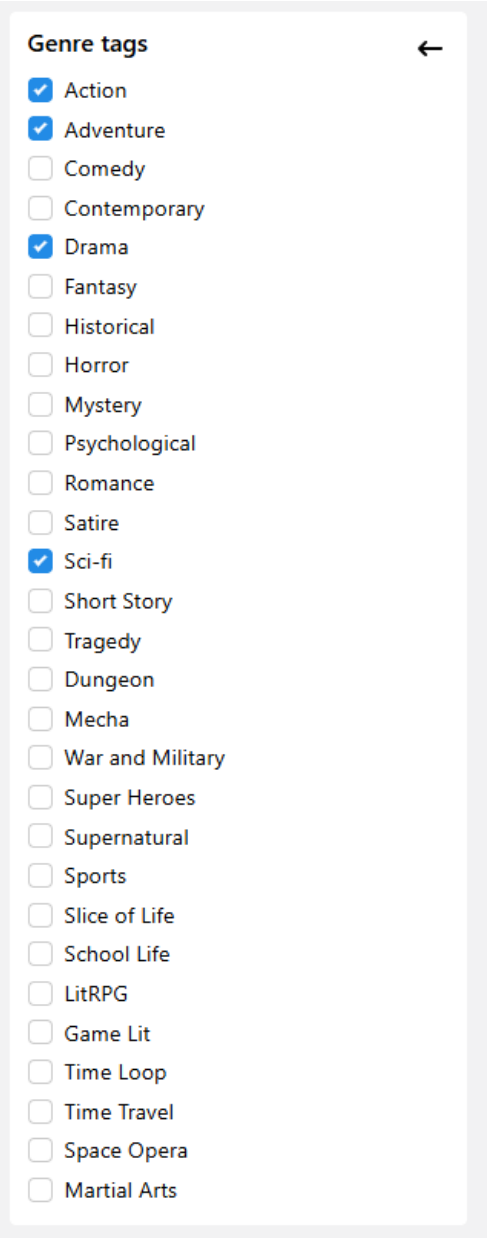

Рисунок 3.18 – Вкладка жанрових тегів

Кнопка «Apply» запускає процес пошуку контенту за обраними фільтрами та передає запит на одну з кінцевих точок REST API.

На рисунку 3.19 наглядно продемонстровано результат успішної пошукової операції. Які саме фільтри були застосовані можна подивитися у минулих двох зображеннях.

Кнопка «Reset» прибирає усі обрані опції разом з текстом компоненту пошуку за назвою. Незмінним лише залишаються обрані значення сортування.

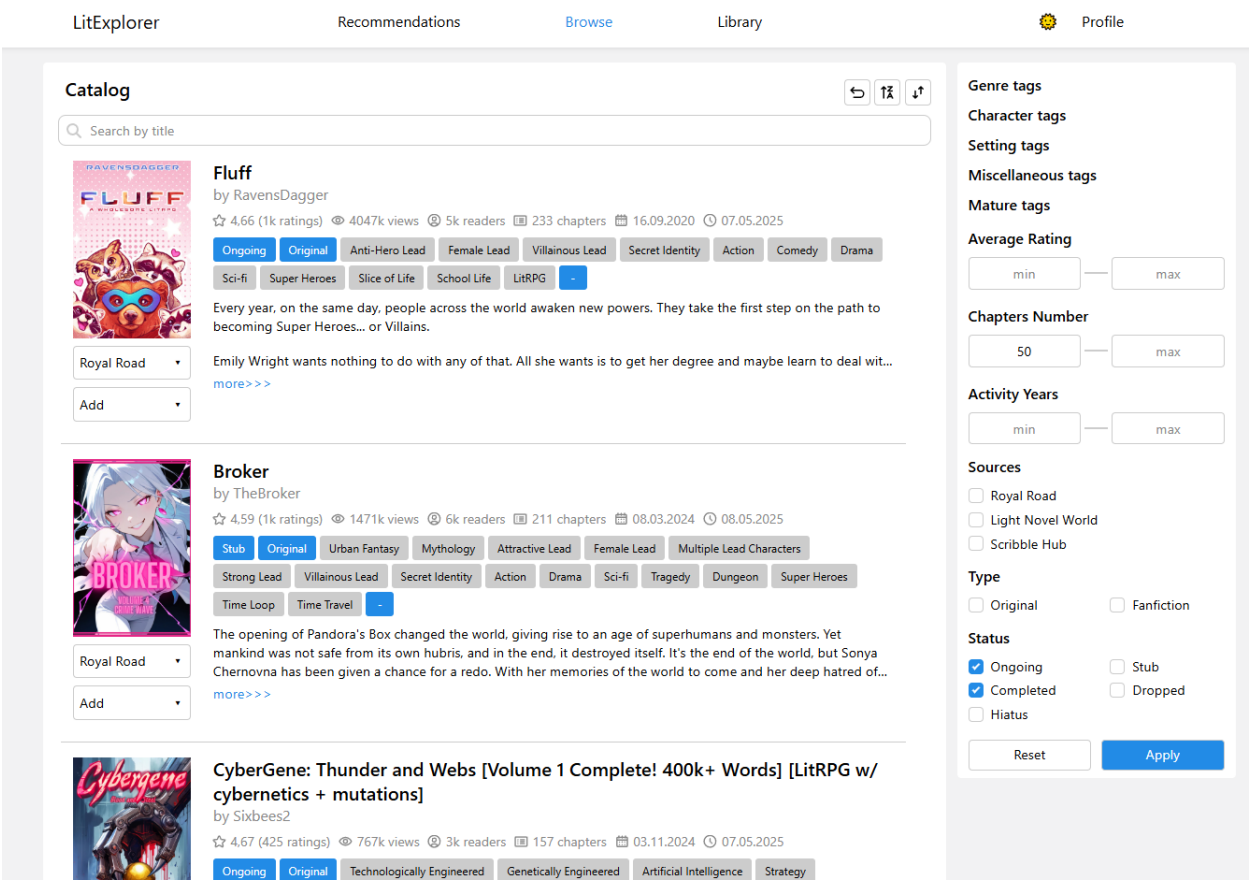

Рисунок 3.19 – Результати успішного пошуку текстів за обраними фільтрами

3.4.3 Сторінка персональної бібліотеки

Бібліотека авторизованого користувача зберігає в собі усі книги з присвяченим особливим статусом. Їх всього 5 видів і більшість з них стосується читацького прогресу. На таблиці 3.1 перелічено ці типи та що вони позначають. Читач також може видалити зі своєї бібліотеки будь-який твір, що не сподобався.

Таблиця 3.1 – Перелік бібліотечних станів книг

Статус	**Опис**
Reading	Клієнт активно читає твір
Favorite	Обраний авторський текст являється одним із найулюбленіших творів, що читав користувач

Продовження таблиці 3.1

Статус	Опис
Planning	Книга лежить на полиці та чекає свого часу
Completed	Читач повністю прочитав весь твір
Dropped	Читач з персональних причин перестав активно читати та «забросив» книгу на дальню полицю, але не захотів видаляти її з бібліотеки

Структурно вебсторінка «Library» (рис. 3.20) майже нічим не відрізняється від сторінки рекомендацій, окрім елемента навігації бібліотеки знизу та інших елементів в панелі фільтрації поверх списку книг. Читач може сортувати твори за одним із варіантів бібліотечного стану (рис. 3.21).

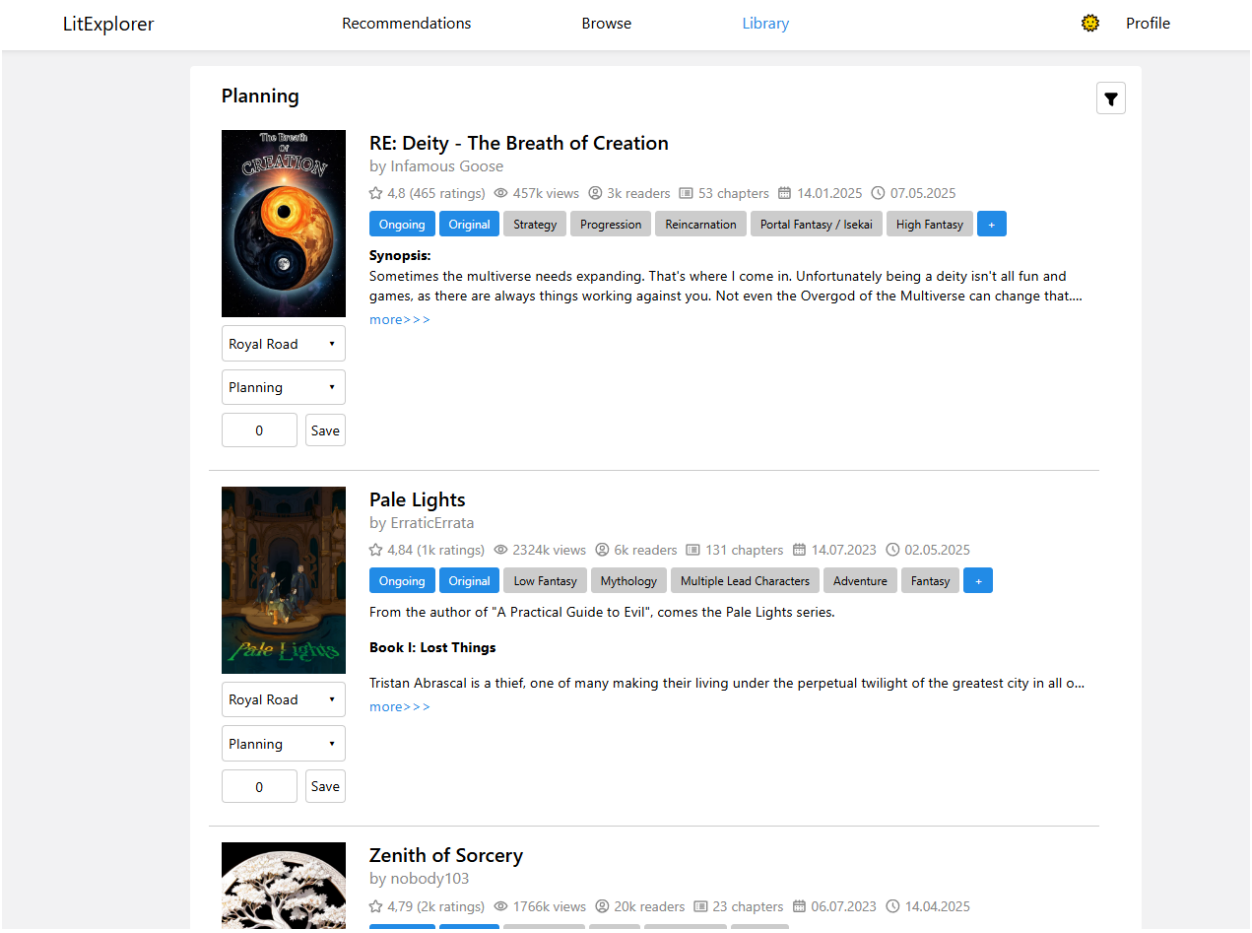

Рисунок 3.20 – Інтерфейс сторінки персональної бібліотеки

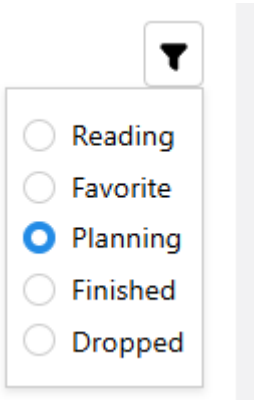

Рисунок 3.21 – Опції сортування за бібліотечним статусом

Будь-які зміни в книжковому статусі буде відразу відображено в поточному списку книг.

Модуль користувацького сховища літератури доступний лише для авторизованих читачів. Посилання на вебсторінку буде сховане для анонімного клієнта.

3.4.4 Компонент книги

Список літератури являється головним елементом клієнтського інтерфейсу. Усі три великі сторінки вебсайту будують свій функціонал коло даного об’єкта. Лист в свою чергу містить в собі максимум 12 компонентів, кожен з яких описує сутність свого твору.

На рисунку 3.22 зображено приклад Razor компоненту BookRow. Назва, зображення обкладинки книги, статистика, опис та багато іншої інформації візуалізовано для полегшення процесу пошуку та аналізу потенційних кандидатів додавання до персональної бібліотеки.

Перша кнопка з меню під зображенням обкладинки, дозволяє змінити джерело виведеної інформації, адаптуючи дані під інший вебресурс. Книга в базі даних – абстрактна сутність, що охоплює усі можливі твори з однаковим ім’ям. Автор може публікувати свій текст на декількох вебсайтах, або

література має спільне лише назву. Статистика, опис, теги, зображення обкладинки – усі ці дані можуть бути різними між багатьох вебплатформ.

Друга кнопка дозволяє змінювати статус книги в рамках персональної бібліотеки: додавання, редагування статусу читання або видалення з користувацького сховища.

Рисунок 3.22 – Приклад компоненту, що представляє авторський текст

3.4.5 Сторінка авторизації

Користувач, що зайшов у систему має доступ до двох ключових функцій платформи: керування персональною бібліотекою та можливість отримати персональні рекомендації.

Користувач, що зайшов у систему має доступ до двох ключових функцій платформи: керування персональною бібліотекою та можливість отримати персональні рекомендації. Для того аби успішно авторизуватися клієнт повинен зайти на відповідну сторінку (рис. 3.23) та ввести свою

електрону адресу з паролем. Перейти до цього інтерфейсу можливо через посилання у верхній правій частині вебсайту, поруч з кнопкою зміни теми.

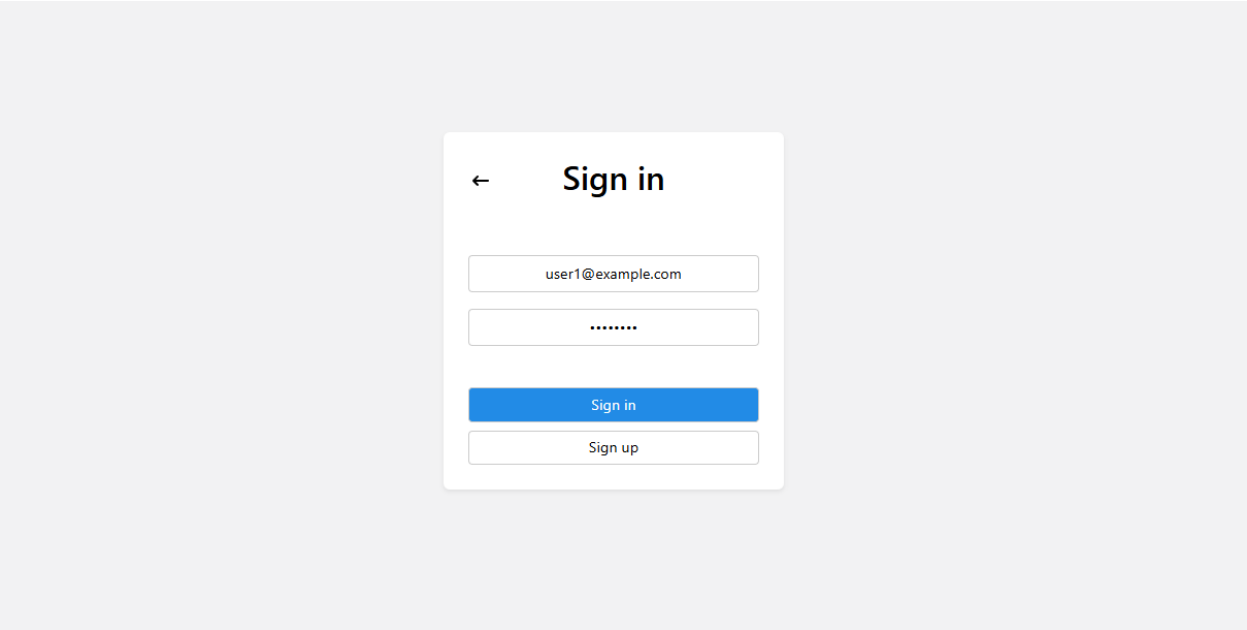

Рисунок 3.23 – Сторінка авторизації

В разі якихось помилок серверу, або неправильності введених даних, буде виведено модальне вікно (рис. 3.24) з відповідним повідомленням. Користувачу потрібно буде повторити процес або зробити зміни в персональній інформації.

Рисунок 3.24 – Модальне вікно з повідомленням про помилку авторизації

Після успішної авторизації клієнта буде перенесено до сторінки каталогу, а посилання в правому верхньому куту вебсайту зміниться на «Profile» та при натисканні буде з’являтися панель (рис. 3.25) з електронною адресою користувача та кнопкою виходу з акаунту.

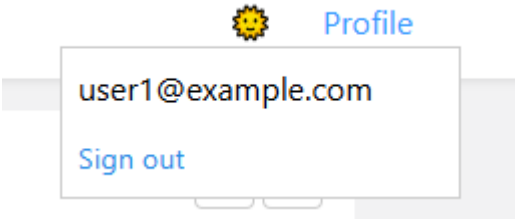

Рисунок 3.25 – Панель профілю авторизованого користувача

Вся інформація про поточний клієнтський сеанс зберігається у локальному сховищі вебсайту. Платформа може зчитувати дані при кожному запуску вебресурса, тому не потрібно буде перезаходити в акаунт кожний раз після перезапуску комп’ютера або браузера.

3.4.6 Сторінка реєстрації

Модуль реєстрації (рис. 3.26) фактично знаходиться в тому ж місці що й модуль авторизації. Аби перейти до нього, потрібно натиснути на саму нижню кнопку «Sign up». Для того щоб перейти до авторизації треба лише активувати кнопку у верхньому лівому куту у вигляді стрілки.

В разі якоїсь помилки або неточності даних також буде з’являтись модальне вікно с повідомленням.

Створений користувачем пароль перетворено в спеціальну хеш строку за допомогою BCrypt технології. Усі операції з паролями проводяться за допомогою цієї бібліотеки.

Застосована у проєкті бібліотека BCrypt.Net-Next забезпечує кросплатформну реалізацію алгоритму bcrypt, побудованого на шифрі Blowfish. Під час хешування до пароля автоматично додається 16-байтова випадкова «сіль», а значення work factor вшивається у сам хеш, що дозволяє збільшувати обчислювальну складність без повторного перехешування вже збережених паролів.

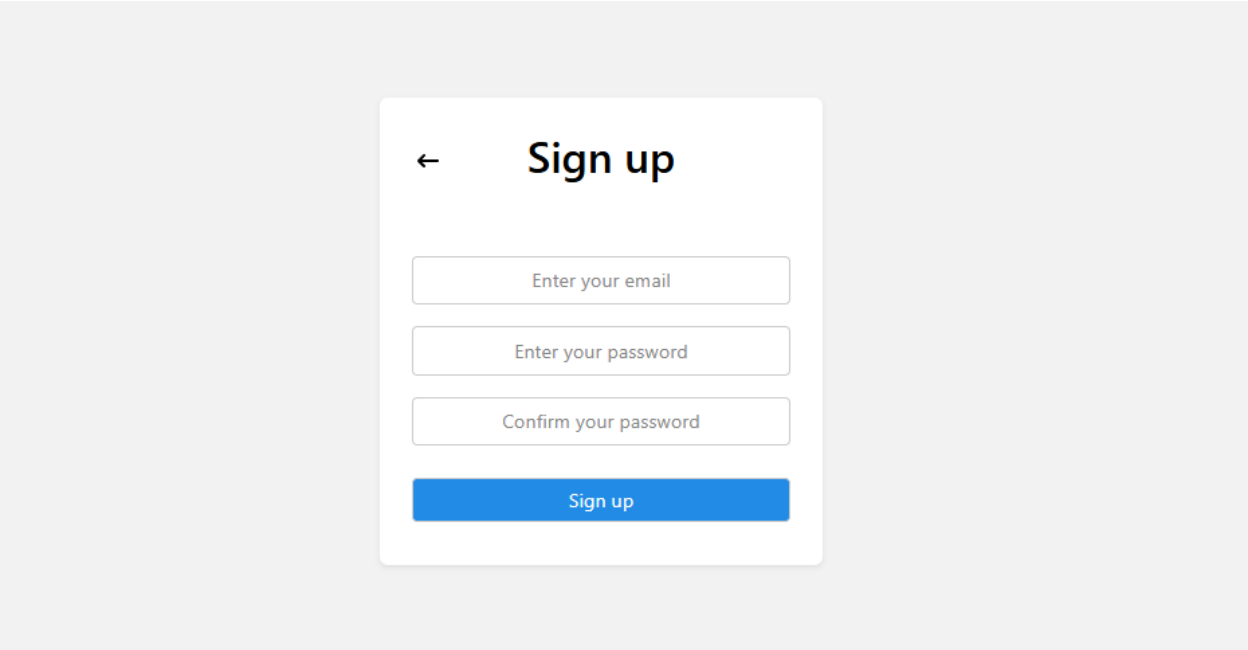

Рисунок 3.26 – Сторінка реєстрації

3.4.7 Реалізація нічного режиму

Майже кожен вебсайт в інтернеті має свій темний режим – зміни в кольоровій палітрі інтерфейсу заради більш сприятливого сприйняття дизайну користувачем в оточенні з низьким рівнем освітлення. Яскраві кольори, частіше білих відтінків, сильно дратують та визивають сухість очей у нічний час.

Клієнтський інтерфейс платформи LitExplorer також має реалізовану опцію зміни режиму відображення UI-елементів. Користувачу лише потрібно натиснути на відповідну кнопку у верхньому правому куту, поруч з посиланням на профіль. Зображення перемикача змінюється в залежності від поточного обраного стану.

Значення режиму зберігається так само як і інформація про поточний авторизований акаунт – за допомогою локального сховища (рис. 3.27). Дані залишаються навіть після перезапуску комп’ютера або браузера.

На рисунках 3.28 та 3.29 зображено дві вебсторінки з увімкненим нічним режимом. Головні білі кольори змінюються на відтінки чорного,

текст стає контрастним білим. Незміненим залишаються лише кольори різних виділених інтерактивних компонентів інтерфейсу, такі як кнопки, елементи текстового вводу, модальні вікна. Ці відтінки синього та червоного являються основним ядром усієї стилістичної архітектури.

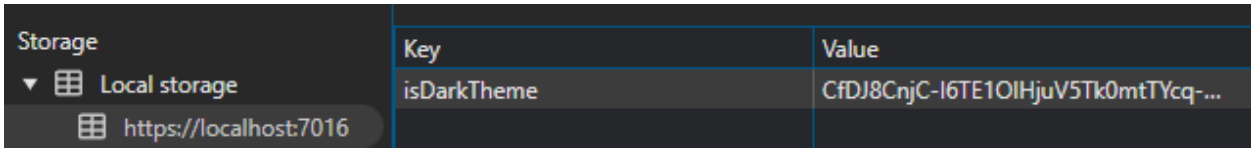

Рисунок 3.27– Локальне сховище застосунку Blazor WebApp

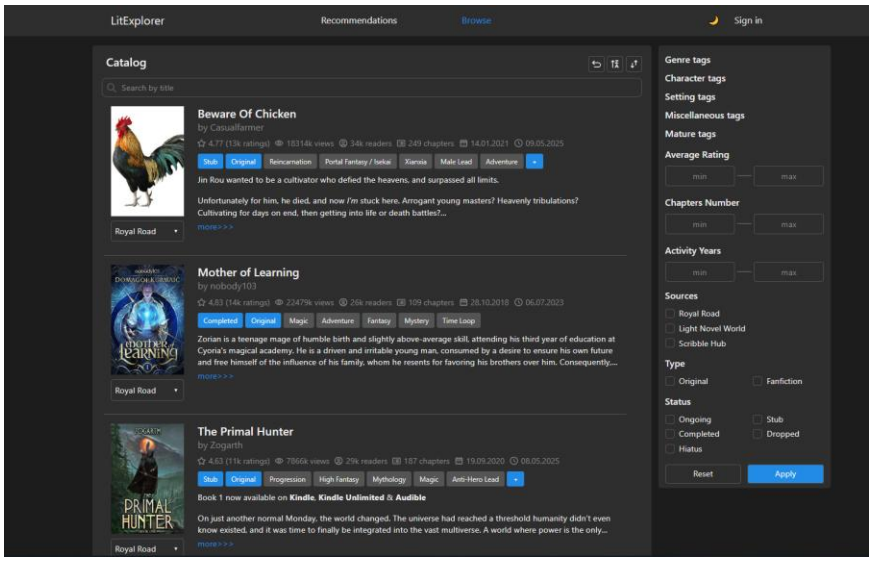

Рисунок 3.28 – Сторінка «Browse» у темному стилі

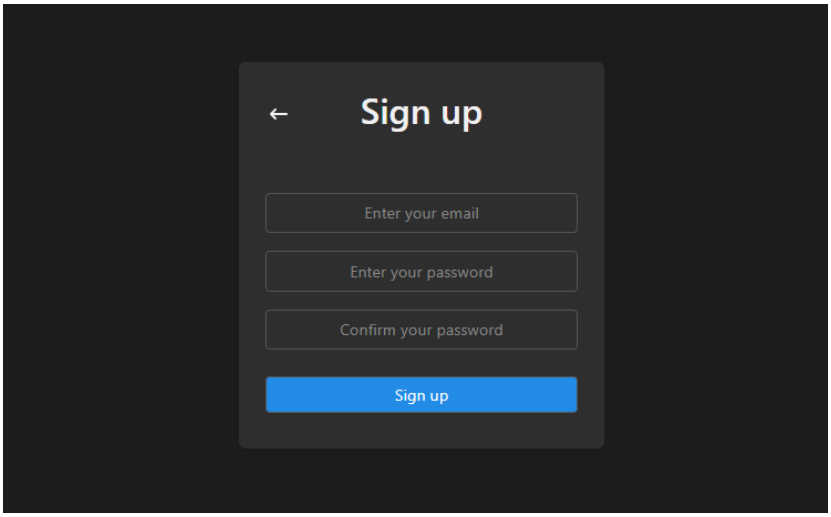

Рисунок 3.29 – Сторінка «SignUp» у темному стилі

ВИСНОВКИ

У рамках кваліфікаційної роботи була розроблена і реалізована платформа об’єднаного пошуку авторських текстів. Вебсервіс представляє собою точку доступу до безкоштовних веброманів, що об’єднує різні літературні ресурси, надає зручний каталог із розширеним фільтруванням та зберігає читацький прогрес у персональній бібліотеці. Платформа вирішує проблему розділення ринку онлайнової прози й підвищує доступність сучасної авторської літератури для широкого кола читачів.

Архітектура вебплатформи була розроблена на основі трьох головних шарів. Перший – шар збирання та нормалізації метаданих книжок з різноманітних оригінальних вебресурсів. Другий – шар централізованого REST API, який дозволяє працювати з головною базою даних зовнішнім клієнтам. Третій – шар клієнтського інтерфейсу на основі Blazor забезпечує інтерактивний пошук, рекомендації та керування профілем читача. Така структура системи гарантує масштабованість, гнучкість розгортання та подальше розширення функціоналу.

Була реалізована контент-орієнтована рекомендаційна система на основі ML.NET, що надає клієнтам літературу, яка базується на основі векторних ознак опису, тегів, персональної статистики читання та вмістом користувацької бібліотеки. Розподіл обчислень на офлайн-підготовку книжкових «відбитків» та онлайн-розрахунок схожості дав змогу досягти майже миттєвого часу відповіді й легкого масштабування разом із зростанням каталогу.

Практична цінність цієї кваліфікаційної роботи полягає у спрощенні доступу до великого масиву безкоштовних авторських текстів, підвищенні зручності навігації серед численної кількості творів різного походження, створення єдиного читацького профілю. Застосунок надає доступ до цілісного читачського середовища, де користувачі можуть відкривати нові романи, відстежувати власний прогрес і отримувати релевантні рекомендації

без необхідності переходити між численними платформами та створювати нові акаунти.

Стрімкий розвиток ІТ-індустрії дозволяє застосувати ідеї з різних сфер професії задля розширення кругозору та досвіду, сприяючи генерації нових варіантів рішення проблем. Багато різних наукових джерел [19–36] було використано у створені даної кваліфікаційної роботи.

Результати роботи апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [17].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Benefits and Challenges of Federated Search. URL: https://www.algolia.com/blog/product/federated-search-benefits-and-challenges (дата звернення 16.04.2025).

2. Federated search. URL: https://en.wikipedia.org/wiki/Federated_search (дата звернення 09.04.2025).

3. eBook Market Outlook for 2024 to 2034. URL: https://www.futuremarketinsights.com/reports/global-ebook-market (дата звернення 09.04.2025).

4. Federated Search Tools: Query All Data and Save Costs. URL: https://gurucul.com/blog/federated-search-tools-query-all-data-and-save-costs/ (дата звернення 12.04.2025).

5. Project Gutenberg. URL: https://www.gutenberg.org (дата звернення 20.04.2025).

6. Internet Archive’s Modern Book Collection Now Tops 2 Million Volumes. URL: https://blog.archive.org/2021/02/03/internet-archives-modern-book-collection-now-tops-2-million-volumes (дата звернення 20.04.2025).

7. lubimyczytac.pl Competitors - Top Sites Like lubimyczytac.pl. URL: https://www.similarweb.com/website/lubimyczytac.pl/competitors/ (дата звернення 02.05.2025).

8. Wattpad Premium Picks. URL: https://www.wattpad.com/catalog/premiumpicks (дата звернення 01.05.2025).

9. webnovel.com vs scribblehub.com Traffic Comparison. URL: https://www.similarweb.com/website/webnovel.com/vs/scribblehub.com/#overview (дата звернення 02.05.2025).

10. lightnovelworld.com Competitors - Top Sites Like lightnovelworld.com. URL: https://www.similarweb.com/website/lightnovelworld.com/competitors (дата звернення 02.05.2025).

11. Inside Tapas Entertainment’s Plans for Webtoons, Webnovels, and Publishing. URL: https://www.publishersweekly.com/pw/by-topic/digital/Apps/article/95453-inside-tapas-entertainment-s-plans-for-webtoons-web-novels-and-publishing.html (дата звернення 11.04.2025).

12. WuxiaWorld Karma shop. URL: https://www.wuxiaworld.com/manage/subscriptions/karma (дата звернення 09.04.2025).

13. Library Genesis. URL: https://en.wikipedia.org/wiki/Library_Genesis (дата звернення 17.04.2025).

14. Anna's Archive. URL: https://en.wikipedia.org/wiki/Anna%27s_Archive (дата звернення 17.04.2025).

15. ASP.NET Core Blazor hosting models. URL: https://learn.microsoft.com/en-us/aspnet/core/blazor/hosting-models (дата звернення 26.04.2025).

16. Tooling for ASP.NET Core Blazor. URL: https://learn.microsoft.com/en-us/aspnet/core/blazor/tooling?view=aspnetcore-9.0&pivots=vs (дата звернення 26.04.2025).

17. Верепа Д.С. (2025) дослідження роботи алгоритму ML.NET для обробки користувацьких рекомендацій. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.)*. Харків: ХНУРЕ, 2025. Т.7. С. 26-28.

18. Recommendation System. URL: https://www.nvidia.com/en-us/glossary/recommendation-system (дата звернення 03.05.2025).

19. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв’язку. Збірник наукових праць*, *5*(57).

20. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

21. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

22. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research*, 7(7), (pp. 25-36).

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

24. Gorokhovatskyi, V., Tvoroshenko, I., & Olena, Y. (2024). Transforming image descriptions as a set of descriptors to construct classification features. *Indonesian Journal of Electrical Engineering and Computer Scienc*, 33 (1), (pp. 113-125)

25. Gorokhovatskyi , V., Chmutov , Y., Tvoroshenko , I., & Kobylin , O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12.

26. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 97–107.

27. Vechirska, I., Kobylin, O., Prokopiev , S., Vechirska, A., & Kucherenko, M. (2022). Building a logical network for solving the problem of car rental by means algebra of finite predicates. Computer Systems and Information Technologies, (2), 78–87.

28. Творошенко, І.С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. *Харків: ХНУРЕ*.

29. Гороховатський, В.О., & Творошенко, І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

30. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

31. Gorokhovatskyi, V.O., Tvoroshenko, I.S., and Peredrii O.O. (2020) Image classification method modification based on model of logic processing of bit description weights vector, *Telecommunications and Radio Engineering*, 79(1), pp. 59-69.

32. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

33. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.

34. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

35. Yakovleva, O., & Nikolaieva, K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4(4), 89-101.

36. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).