МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ СТВОРЕННЯ ТА УПРАВЛІННЯ АВТОРСЬКИМИ ТЕКСТАМИ З МОЖЛИВІСТЮ ПУБЛІКАЦІЇ»**

Виконала: студентка 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Ю.А. Перегородо
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Миронова Н.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри програмної інженерії, доцент, к.т.н Лимаренко Ю.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки

(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент
Пшенична О.С.
(підпис)

" 08 " грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Перегородо Юлії Андріївні
(прізвище, ім'я та по-батькові)

1. Тема роботи Проектування та розробка вебплатформи для створення та управління авторськими текстами з можливістю публікації

керівник роботи Миронова Наталя Олексіївна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	16.01.2026	
2.	Збір вихідних даних.	28.02.2026	
3.	Обробка методичних та теоретичних джерел.	11.03.2026	
4.	Розробка першого та другого розділу.	15.04.2026	
5.	Розробка третього розділу.	12.05.2026	
6.	Оформлення та нормоконтроль кваліфікаційної роботи бакалавра.	05.06.2026	
7.	Захист кваліфікаційної роботи.	19.06.2026	

Студент _____ Ю.А. Перегородо
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.О. Миронова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проектування та розробка вебплатформи для створення та управління авторськими текстами з можливістю публікації»: 68 с., 12 рис., 4 табл., 20 джерел, 1 додаток.

АВТОРСЬКІ ТЕКСТИ, ВЕБПЛАТФОРМА, МОДЕРАЦІЯ КОНТЕНТУ, ПУБЛІКАЦІЯ ІСТОРІЙ, РЕКОМЕНДАЦІЙНА СИСТЕМА, FLASK, MYSQL, REACT, TF-IDF.

Об’єкт дослідження – процеси створення, публікації та читання авторських текстів у вебсередовищі.

Предмет дослідження – вебплатформа для роботи з авторськими текстами, що містить засоби публікації, взаємодії користувачів, модерації контенту, відстеження читацької активності та формування персоналізованих рекомендацій.

Мета роботи: розробити вебзастосунок, який надає авторам зручні інструменти для створення й публікації текстів, а читачам – можливість шукати, переглядати та оцінювати контент, взаємодіяти з іншими користувачами й отримувати добірки історій відповідно до власних інтересів.

Методи дослідження: аналіз предметної області та наявних платформ для публікації текстів, порівняння функціональних можливостей аналогів, проєктування структури вебзастосунку і бази даних, програмна реалізація клієнтської та серверної частин, тестування працездатності розробленої системи.

SUMMARY

Bachelor’s Qualifying Theses “Design and Development of a Web Platform for Creating and Managing Author's Texts with the Possibility of Publication”: 68 pages, 12 figures, 4 tables, 20 references, 1 appendix.

AUTHORIAL TEXTS, CONTENT MODERATION, FLASK, MYSQL, PUBLICATION OF STORIES, REACT, RECOMMENDATION SYSTEM, TF-IDF, WEB PLATFORM.

Object of research – the processes of creating, publishing, and reading authorial texts in a web environment.

Subject of research – a web platform for working with authorial texts that includes tools for publication, user interaction, content moderation, tracking reading activity, and generating personalized recommendations.

Aim of the study: to develop a web application that provides authors with convenient tools for creating and publishing texts, and readers with the ability to search for, view, and evaluate content, interact with other users, and receive selections of stories according to their interests.

Methods of research: analysis of the subject area and existing text publication platforms, comparison of the functional capabilities of analogues, design of the web application structure and database, software implementation of the client and server parts, and testing of the developed system.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 7
1 Аналіз предметної області та вибір технологі 10
1.1 Актуальність створення платформи для публікації авторських проектів 10
1.2 Аналіз існуючих веб застосунків для публікації авторських текстів 11
1.3 Вимоги до вебзастосунку 14
1.4 Обґрунтування вибору технологій 17
2 Проєктування та розробка вебплатформи 20
2.1 Use Case діаграма та опис 20
2.2 Архітектура системи 24
2.3 Проєктування бази даних 27
2.4 Розробка серверної частини 31
2.5 Розробка клієнтської частини 37
2.6 Алгоритм рекомендацій 46
3 Тестування програмного продукту 50
3.1 Функціональне тестування 50
3.2 Оцінка продуктивності та стабільності роботи системи 54
3.3 Оцінка ефективності рекомендаційної системи 56
Висновки 60
Перелік посилань 63
Додаток А Приклад коду рекомендацій 635

ВСТУП

Упродовж останніх років вебсервіси стали одним із головних середовищ для творчого самовираження. Автори дедалі частіше публікують власні твори не в друкованому форматі, а на онлайн-платформах, де текст можна швидко розмістити, редагувати, доповнювати новими розділами та відразу отримувати реакцію читачів [1]. Такий формат особливо важливий для початківців, які потребують доступного простору для представлення своїх робіт і поступового формування аудиторії.

Вебплатформи для авторських текстів поєднують дві важливі складові. З одного боку, вони надають авторам інструменти для створення й публікації історій, романів, фанфіків, поезії та інших видів текстового контенту. З іншого боку, вони забезпечують читачам зручний доступ до творів, можливість взаємодіяти з авторами, залишати коментарі, зберігати улюблені матеріали та стежити за оновленнями. Саме тому подібні системи вже давно перестали бути простими електронними бібліотеками й перетворилися на повноцінні цифрові спільноти [1].

Попри наявність великої кількості сервісів для онлайн-публікацій, проблема створення зручної та функціональної платформи залишається актуальною. Частина існуючих рішень перевантажена зайвими можливостями, інші недостатньо зручні для нових користувачів або не приділяють належної уваги персоналізації, контролю якості контенту та взаємодії між учасниками спільноти. Крім того, для сучасного користувача важливо не лише мати можливість опублікувати текст, а й отримати зручний профіль, статистику, систему сповіщень, рекомендації відповідно до власних уподобань та інструменти для комфортного читання.

Актуальність теми дипломної роботи зумовлена потребою у розробці вебплатформи, яка поєднує функції створення, публікації, перегляду та обговорення авторських текстів у межах єдиної системи. Окрему цінність має

реалізація персоналізованого добору контенту, що допомагає читачам швидше знаходити цікаві твори, а авторам – отримувати додаткову видимість серед потенційної аудиторії. Використання машинного навчання в рекомендаційному модулі підвищує практичну значущість системи та робить її ближчою до сучасних цифрових сервісів.

Метою дипломної роботи є розробка вебплатформи для створення й публікації авторських текстів, яка забезпечує зручну взаємодію між авторами та читачами, підтримує керування контентом, персоналізовані рекомендації, комунікацію користувачів і базові механізми модерації.

Для досягнення поставленої мети необхідно виконати такі завдання:

- дослідити особливості предметної області та проаналізувати наявні вебсервіси для публікації авторських текстів;
- визначити функціональні й нефункціональні вимоги до розроблюваної системи;
- спроєктувати структуру вебзастосунку та логічну модель бази даних;
- реалізувати серверну частину для обробки користувацьких запитів, авторизації, роботи з текстами та пов’язаними сутностями;
- розробити клієнтський інтерфейс для авторів, читачів і модераторів;
- забезпечити можливість створення історій, додавання розділів, редагування публікацій і керування статусом матеріалів;
- реалізувати функції взаємодії користувачів із контентом: коментарі, відповіді, вподобання, закладки, підписки та сповіщення;
- передбачити засоби відстеження історії читання й прогресу користувача;
- реалізувати систему пошуку, тематичної навігації та персоналізованих рекомендацій із використанням машинного навчання;
- впровадити базові механізми модерації, зокрема подання скарг, опрацювання звернень і блокування порушників;

– провести перевірку працездатності розробленого вебзастосунку та оцінити коректність реалізованого функціоналу.

Об’єктом дослідження є процеси створення, публікації та споживання авторського текстового контенту у вебсередовищі.

Предметом дослідження є методи й програмні засоби розробки вебплатформи для авторських текстів, зокрема механізми керування контентом, взаємодії користувачів, персоналізації та модерації.

Під час виконання роботи використано методи аналізу предметної області, проєктування програмних систем, моделювання баз даних, клієнт-серверного підходу до побудови вебзастосунків, об’єктно-орієнтованого програмування, а також методи тестування програмного забезпечення. Для реалізації рекомендаційного компонента застосовано підходи, орієнтовані на інтелектуальний добір контенту з урахуванням характеристик творів і користувацької активності.

Практичне значення роботи полягає у створенні повноцінної вебплатформи, яку можна використовувати як основу для онлайн-сервісу публікації авторських текстів. Застосунок створений не лише для публікації текстів. Він об’єднує все необхідне для зручної взаємодії авторів і читачів: особисті профілі, історії та розділи, рекомендації, коментарі, підписки, сповіщення і систему модерації. Це робить систему придатною для подальшого розвитку та розширення функціональних можливостей.

Структурно дипломна робота складається зі вступу, трьох розділів, висновків і списку використаних джерел. У першому розділі розглянуто предметну область, визначено основні проблеми та проаналізовано подібні вебсервіси. У другому розділі описано проєктування та реалізацію розробленої платформи, її архітектуру, базу даних і ключові функціональні модулі. Третій розділ присвячено перевірці працездатності системи, аналізу отриманих результатів і можливим напрямам подальшого розвитку проєкту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ

1.1 Актуальність створення платформи для публікації авторських проектів

Публікація авторських текстів є важливою частиною сучасного цифрового середовища. Інтернет надає авторам можливість ділитися власною творчістю з широкою аудиторією, а читачам швидко знаходити новий й цікавий контент. Для багатьох людей написання текстів починається як захоплення, багато людей просто не можуть знайти те що їм сподобається й починають писати самостійно. Таке хобі з часом може перетворитися на форму самовираження, професійного розвитку або спосіб знайти однодумцями та можливо завести друзів.

Авторські історії, художні твори, есе, статті та інші текстові матеріали виконують не лише розважальну функцію. Вони дають змогу порушувати суспільно важливі, культурні та моральні питання, передавати особистий досвід, формувати дискусію навколо різних тем. Завдяки відкритому доступу до таких матеріалів читачі можуть знайомитися з новими поглядами, відкривати невідомих авторів і долучатися до спільнот за інтересами.

Платформи для публікації текстів також відіграють помітну роль у сфері дозвілля. Користувачі читають твори у зручний для себе час, стежать за продовженням історій, додають матеріали до закладок, залишають коментарі та взаємодіють з авторами. Водночас подібні сервіси можуть мотивувати читачів спробувати себе у творчості й перейти від споживання контенту до його створення.

Уваги заслуговують й культурне значення таких вебресурсів. Вони підтримують інтерес до читання, сприяють розвитку українськомовного контенту та допомагають формувати активне літературне середовище. Для авторів це можливість заявити про себе без складного шляху знайти

видавництво, яке погодиться опублікувати історію. Для видавців і редакторів такі платформи можуть бути додатковим джерелом спостереження за реакцією читачів на окремі жанри, твори та авторські стилі та пошук нових зірок.

Український сегмент онлайн платфор уже має невелику кількість сервісів. Одним із відомих в кругах письменників, прикладом є платформа «Аркуш», яка позиціонується як україномовний літературний простір для авторів і читачів. Водночас чимало користувачів звертаються і до міжнародних платформ, хоча вони не завжди враховують специфіку локальної аудиторії, мовне середовище та потребу в сервісах, орієнтованих саме на український культурний простір [2].

Усе це підтверджує актуальність розробки вебплатформи, яка поєднує створення, публікацію та читання авторських текстів із сучасними засобами взаємодії між користувачами. Такий застосунок має бути зручним для авторів і читачів, підтримувати роботу з профілями, текстами, коментарями, вподобаннями, закладками, сповіщеннями та персоналізованими рекомендаціями. Саме комплексність цих можливостей підвищує практичну цінність розроблюваної системи.

1.2 Аналіз існуючих веб застосунків для публікації авторських текстів

З розвитком інтернету з’явилося чимало платформ, на яких користувачі можуть публікувати власні тексти, знаходити читачів і поступово розвивати свою аудиторію. Такі сервіси особливо важливі для авторів-початківців, адже дають змогу представити свої роботи без звернення до видавництв або складної процедури відбору. Аналіз уже наявних рішень допомагає визначити сильні сторони подібних платформ, помітити їхні обмеження та врахувати це під час розробки власного вебзастосунку.

В українському сегменті платформ для публікації авторських текстів поки що небагато. Одним із найпомітніших прикладів є «Аркуш», який позиціонується як простір для читачів і письменників та орієнтується на україномовний літературний контент. Платформа підтримує розміщення творів, знайомство з авторами та читання матеріалів різних жанрів. Водночас за своїм загальним підходом вона залишається доволі стриманою за функціональністю, тому під час створення нового рішення доцільно приділити більше уваги персоналізації, зручній взаємодії користувачів і сучасному інтерфейсу [2].

Однією з найвідоміших міжнародних платформ є Wattpad. Сервіс об’єднує велику спільноту читачів і авторів, підтримує публікацію оригінальних історій, коментарі, читацькі списки, сповіщення про оновлення та інструменти для пошуку творів за жанрами. Для автора це зручне середовище, у якому можна швидко отримати перші реакції на текст. Водночас через дуже велику кількість матеріалів новим публікаціям складніше привернути увагу, а значна частина роботи сервісу пов’язана з комерційними можливостями та платними функціями [3].

Ще одним помітним сервісом є Webnovel. Платформа зосереджується переважно на серіалізованих творах, які виходять частинами, та підтримує авторів, що працюють із довгими сюжетними форматами. Окрему роль у цій системі відіграє монетизація контенту, зокрема співпраця з авторами та отримання доходу від популярних творів. Такий підхід є ефективним для комерційних літературних проєктів, проте він більше підходить для авторів, які вже орієнтуються на регулярний випуск великих за обсягом історій [4][5].

Платформа Royal Road також спеціалізується на веброманах і фанфікшені. Вона має активну спільноту читачів, які оцінюють твори, залишають відгуки та стежать за розвитком історій. Найчастіше сервіс асоціюється з довгими художніми серіями, зокрема у жанрах фентезі, пригодницької прози та наукової фантастики. Для дипломного проєкту Royal Road є цікавим прикладом платформи, де спільнота й регулярне оновлення

контенту мають велике значення, однак таке рішення більше орієнтоване на певний тип літератури та англомовну аудиторію [6].

Окремо варто розглянути Medium. На відміну від попередніх платформ, цей сервіс більше пов’язаний із публіцистикою, аналітичними матеріалами, есе та тематичними статтями. Автори можуть публікувати власні тексти, об’єднуватися навколо видань за темами та брати участь у партнерській програмі для монетизації матеріалів. Через це Medium менш доречний як орієнтир для художньої платформи, однак його підхід до подачі тексту, зручності читання та організації публікацій є корисним для аналізу [7]. Основні недоліки популярних платформ для публікації авторських текстів наведено у табл. 1.1.

Таблиця 1.1 – Основні недоліки популярних платформ для публікації авторських текстів

Назва платформи	**Основні недоліки**
Аркуш	Застарілий інтерфейс, періодичні технічні помилки, обмежений функціонал взаємодії
Wattpad	Висока конкуренція серед авторів, велика кількість реклами, складність просування нових текстів
Webnovel	Часткова платність контенту, орієнтація на комерційні серіали, бар’єри для нових авторів
Royal Road	Вузька жанрова спрямованість, застарілий дизайн, обмежена локалізація
Medium	Орієнтація переважно на статті та блоги, обмежена підтримка художніх текстів

Отже, кожна з наведених платформ має власну модель розвитку та набір функцій. Одні сервіси більше орієнтовані на літературні спільноти, інші на комерційну публікацію або роботу з короткими публіцистичними матеріалами. Для розроблюваної вебплатформи важливо поєднати

доступність для автора, зручність для читача, україномовне середовище, сучасні інструменти взаємодії та персоналізований добір текстів. Саме поєднання цих характеристик дозволяє обґрунтувати доцільність створення нового застосунку.

1.3 Вимоги до вебзастосунку

Після аналізу наявних платформ для публікації авторських текстів доцільно визначити вимоги до вебзастосунку, що розробляється в межах дипломної роботи. Це важливий етап, оскільки саме вимоги впливають на подальше проєктування архітектури системи, структури бази даних, логіки роботи серверної частини та інтерфейсу користувача. Якщо сформулювати їх нечітко або не врахувати реальні потреби майбутніх користувачів, застосунок може виявитися незручним або неповним за функціональністю.

Розроблювана платформа має бути зрозумілою для двох основних категорій користувачів: тих, хто створює та публікує власні твори, і тих, хто читає ці матеріали та взаємодіє з ними. Водночас один користувач може поєднувати обидві ролі, наприклад, одночасно бути автором історій і читачем інших публікацій. Окремо в системі передбачаються модератор і адміністратор, які відповідають за контроль дотримання правил платформи та опрацювання скарг.

До основних функціональних вимог належить реєстрація та авторизація користувачів за допомогою електронної пошти й пароля. Після входу користувач отримує доступ до особистого профілю, де може змінювати основні дані, завантажувати аватар, переглядати власні історії, вподобані матеріали, закладки, підписки та статистику активності. Для відкритих профілів передбачено відображення базової інформації про автора, опублікованих творів і загальної статистики.

Користувач, який публікує тексти, повинен мати змогу створювати нові історії, додавати до них опис, тип твору, вікове обмеження, жанри й обкладинку. Окремою вимогою є підтримка структури твору у вигляді розділів. Автор може створювати глави, редагувати їх, зберігати як чернетки або публікувати для читачів. Такий підхід зручний для роботи з великими творами, які доповнюються поступово. Також передбачено можливість змінювати статус історії, позначати її як завершену та редагувати вже опубліковані матеріали.

Для читачів важливо, щоб система давала змогу зручно знаходити, відкривати й читати історії та їхні розділи. Користувач повинен мати можливість відкривати сторінку твору, читати його опис, бачити жанри, автора, кількість переглядів, лайків і коментарів, а також переходити до потрібної глави. Усередині сторінки читання необхідна проста навігація між розділами без зайвих дій. Крім того, у застосунку передбачено збереження історії читання та відстеження прогресу, що дає змогу повернутися до твору пізніше.

Окремою вимогою є зручна навігація по великій кількості матеріалів. Система повинна підтримувати пошук за назвою історії та іменем автора, а також добірки, сортування й фільтрацію публікацій за різними характеристиками. Для кращого підбору контенту в застосунку реалізується рекомендаційний модуль із використанням машинного навчання, який формує персоналізовані пропозиції на основі даних про тексти та користувацьку активність.

Важливою частиною платформи є взаємодія між учасниками. Користувачі повинні мати можливість залишати коментарі під розділами, відповідати на них, ставити вподобання, додавати історії до закладок і підписуватися на авторів. Для інформування про нові події передбачено систему сповіщень, наприклад, про коментарі, відповіді, нові підписки, вподобання та появу нових глав.

Для підтримання порядку на платформі необхідні інструменти модерації. Користувач може подати скаргу на історію або коментар. Модератор чи адміністратор має доступ до окремої панелі, де переглядаються звернення, ухвалюються рішення щодо них, видаляється контент із порушеннями та застосовуються обмеження до облікових записів користувачів. Система також повинна підтримувати тимчасове або безстрокове блокування профілю залежно від характеру порушення.

Окрім функціональних можливостей, важливими є нефункціональні вимоги. Інтерфейс застосунку має бути зрозумілим, охайним і логічно організованим. Вебплатформа повинна коректно відображатися на персональних комп’ютерах, планшетах і смартфонах. Для комфортної роботи користувачів важливі швидке завантаження сторінок, стабільність основних функцій і раціональна взаємодія з базою даних.

Окрему увагу необхідно приділити безпеці. Паролі користувачів мають зберігатися в захищеному вигляді, доступ до приватних функцій повинен надаватися лише після авторизації, а дії, пов’язані з редагуванням або видаленням контенту, мають бути доступні тільки його власнику або відповідальним представникам адміністрації. Таке розмежування прав доступу знижує ризик несанкціонованих дій у системі [16].

Важливою вимогою є й зручність редактора текстів. Він повинен підтримувати базове форматування, редагування чернеток і комфортну підготовку глав до публікації. Для автора це один із ключових інструментів, тому його робота має бути простою та передбачуваною.

Отже, сформовані вимоги охоплюють основні сценарії роботи авторів, читачів і модераторів, а також визначають якісні характеристики системи. Вони становлять основу для подальшого проєктування вебплатформи, структури бази даних і реалізації її функціональних модулів.

1.4 Обґрунтування вибору технологій

Під час розробки вебплатформи важливо правильно визначити набір технологій. Від цього залежить зручність реалізації, стабільність роботи системи, можливість подальшого розширення та підтримка коду. Оскільки застосунок має бути доступним через браузер без встановлення додаткових програм, основою було обрано вебтехнології.

Для серверної частини розглядалися Python, Node.js і PHP. У межах цього проєкту було обрано Python, оскільки він має зрозумілий синтаксис, зручний для розробки серверної логіки, а також широкий набір бібліотек для обробки даних і реалізації рекомендаційних механізмів. Це важливо для системи, у якій передбачено персоналізований добір текстів із використанням машинного навчання.

Серед Python-фреймворків доцільно було обрати Flask. Django має широкий набір вбудованих можливостей, однак у межах цього проєкту значна частина з них була б зайвою. Flask є легшим і дозволяє підключати лише ті модулі, які справді потрібні. У розробленій системі він використовується для створення REST API, маршрутизації запитів, обробки даних у форматі JSON і взаємодії з базою даних. Для захисту маршрутів застосовано flask-jwt-extended, для налаштування доступу з клієнтської частини використано flask-cors [8] [13].

Для зберігання даних обрано MySQL. У системі використовується чітко визначена структура сутностей: користувачі, історії, глави, коментарі, лайки, закладки, сповіщення, історія читання та підписки. Такі дані зручно організовувати в реляційній моделі через таблиці та зв’язки між ними. MySQL добре підходить для подібної структури й дозволяє ефективно виконувати основні операції створення, читання, оновлення та видалення записів. На етапі розробки також зручно використовувати phpMyAdmin для перегляду та редагування вмісту бази [11].

Клієнтську частину реалізовано на React із використанням Vite. React дає змогу будувати інтерфейс із незалежних компонентів, що добре відповідає структурі застосунку. Наприклад, окремими компонентами виступають картка історії, блок коментарів, панель сповіщень, банер, сторінка профілю та секції добірок. Це спрощує підтримку коду й повторне використання готових елементів. Vite використовується як інструмент збирання та запуску фронтенду, оскільки забезпечує швидку розробку й зручну роботу з сучасним JavaScript-проєктом [9] [10].

Для реалізації навігації між сторінками підключено react-router-dom. Додатково використовуються допоміжні бібліотеки, зокрема react-select для зручного вибору жанрів і date-fns для роботи з датами. Це дозволяє не витрачати час на створення базових інтерфейсних рішень з нуля та зосередитися на логіці самої платформи.

Авторизація реалізована на основі JWT. Такий підхід зручний для REST API, оскільки сервер не повинен зберігати окрему сесію для кожного користувача. Після входу користувач отримує токен, який надсилається в заголовку запиту й використовується для перевірки доступу до захищених маршрутів. У проєкті токен має обмежений термін дії, після якого потрібно повторно пройти авторизацію [12].

Для написання й редагування глав використано Tiptap. Це редактор форматованого тексту, побудований на основі ProseMirror. Він дозволяє підтримувати базове оформлення матеріалів, зокрема жирний і курсивний текст, підкреслення, закреслення та вирівнювання. Такий редактор зручний для авторів, оскільки дає змогу працювати з текстом у звичному форматі, не використовуючи HTML вручну [14].

Стилізація інтерфейсу виконується за допомогою SCSS. У порівнянні зі звичайним CSS він надає зручніші засоби організації стилів, зокрема вкладені правила, змінні та повторне використання фрагментів оформлення. Це особливо корисно для застосунку з великою кількістю сторінок і компонентів[15].

Архітектурно система побудована як SPA на клієнтській частині та REST API на сервері. Такий підхід дає змогу оновлювати потрібні дані без повного перезавантаження сторінки, що покращує зручність користування. Окремою перевагою є можливість надалі використовувати той самий серверний API не лише для вебверсії, а й для мобільного застосунку. Коротке порівняння вибраних технологій з альтернативами наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння вибраних технологій

Компонент	**Вибрана технологія**	**Альтернативи**	**Причина вибору**
Фронтенд	React + Vite	Vue.js, Angular	Компонентний підхід, зручність розробки, сучасний інструмент збирання
Бекенд	Flask, Python	Django, Node.js, PHP	Гнучкість, простота реалізації REST API, зручність для рекомендаційних механізмів
База даних	MySQL	PostgreSQL, MongoDB	Відповідність реляційній структурі даних, простота роботи
Авторизація	JWT	Серверні сесії	Зручність для API та розмежування доступу
Редактор	Tiptap	TinyMCE, Quill	Гнучке форматування тексту та зручна інтеграція
Стилі	SCSS	CSS, Tailwind CSS	Зручніша організація великих стилів

Отже, обраний стек технологій відповідає задачам дипломного проєкту. Він дозволяє реалізувати зручний вебзастосунок для авторів і читачів, підтримати персоналізовані рекомендації, організувати безпечну авторизацію та забезпечити можливість подальшого розвитку системи.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ

2.1 Use Case діаграма та опис

Щоб чітко показати взаємодію користувачів із вебплатформою, на етапі проєктування було створено Use Case діаграму. Вона дає змогу побачити основні ролі в системі та дії, які доступні кожній із них. Такий спосіб подання є зручним на початковому етапі розробки, оскільки допомагає узгодити майбутній функціонал ще до реалізації окремих модулів.

У розробленому вебзастосунку передбачено чотири ролі: гість, автор, читач і модератор. Автор і читач не є взаємовиключними ролями. Один зареєстрований користувач може одночасно читати чужі тексти, взаємодіяти з ними та публікувати власні роботи. Модератор має розширені права доступу для контролю за дотриманням правил платформи. У межах реалізації системи функції модерації також можуть бути доступні адміністратору.

Діаграму прецедентів наведено на рис. 2.1. Вона відображає основні сценарії роботи із системою та показує, які можливості надаються користувачам залежно від їхньої ролі.

Гість може користуватися відкритою частиною платформи без створення облікового запису. Йому доступний перегляд опублікованих історій, читання глав, пошук за назвою, автором або ключовими словами, перегляд сторінок авторів і списку жанрів. Також гість може бачити коментарі до глав та загальну статистику історій, зокрема кількість переглядів, лайків і коментарів. Водночас він не може залишати власні коментарі, додавати твори до закладок, ставити вподобання чи підписуватися на авторів.

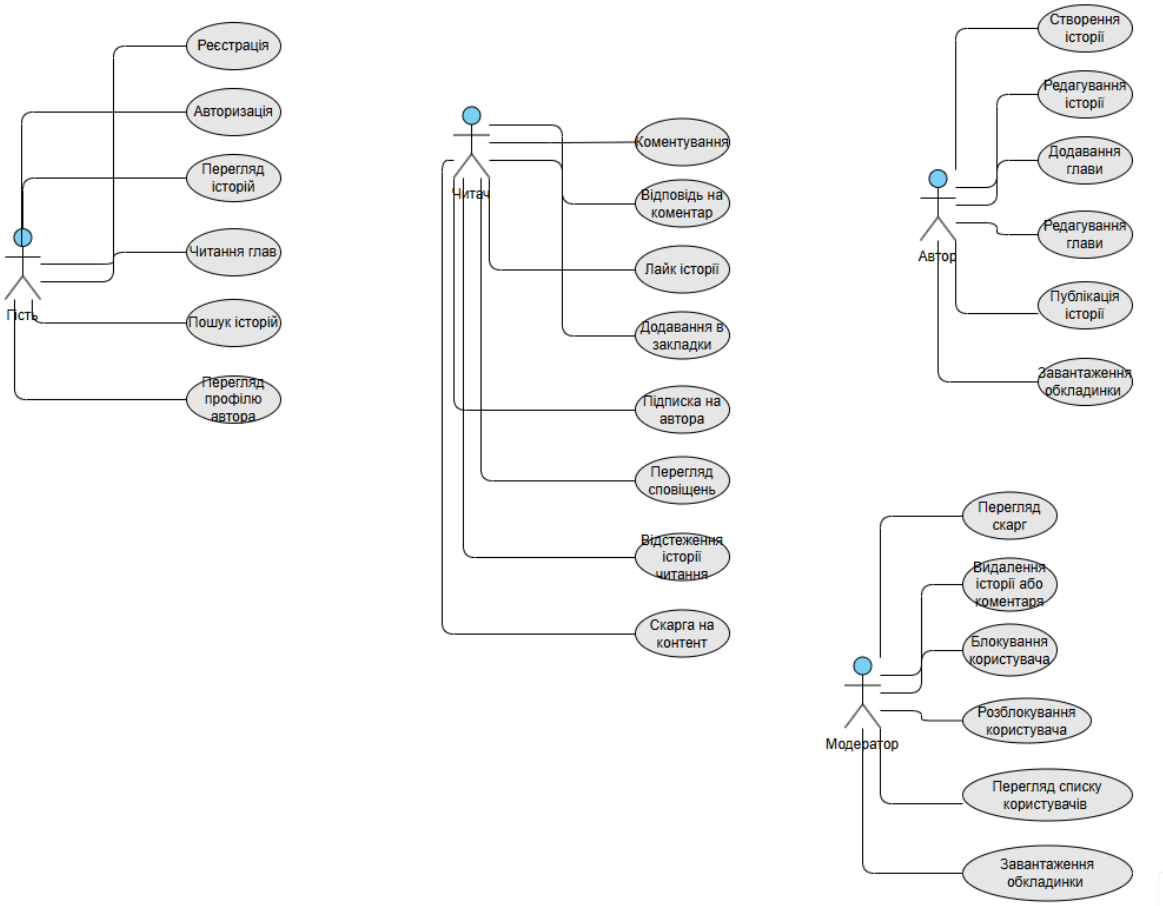


Рисунок 2.1 – Use Case діаграма

Автор отримує доступ до інструментів створення й керування власним контентом. Під час додавання історії він зазначає назву, опис, жанри, тип твору та вікове обмеження. За потреби можна завантажити обкладинку. Кожна історія складається з окремих глав, які створюються через текстовий редактор із підтримкою базового форматування. Автор може зберігати глави як чернетки, публікувати їх пізніше, редагувати вже створені матеріали, оновлювати відомості про історію, змінювати обкладинку та видаляти власні тексти. Якщо роботу завершено, автор може позначити її відповідним статусом.

Читач користується функціями, пов’язаними з переглядом і взаємодією з контентом. Він може ставити лайки історіям, додавати їх до закладок, залишати коментарі під главами та відповідати іншим користувачам. Також читач може підписуватися на авторів і отримувати сповіщення про появу

нових глав. У системі реалізовано відстеження прогресу читання, завдяки чому користувач бачить історії, які переглядав раніше, та відсоток прочитаного. Окремо передбачено можливість подати скаргу на історію або коментар, якщо вміст порушує правила платформи.

Для зареєстрованого користувача також формується персоналізована добірка рекомендацій. Вона враховує його вподобання, закладки та історію читання. Завдяки цьому на головній сторінці можуть відображатися твори, які з більшою ймовірністю відповідають його інтересам.

Модератор працює з адміністративною частиною платформи. У панелі модерації він переглядає скарги, подані користувачами, аналізує проблемний контент і приймає рішення щодо подальших дій. За результатами перевірки модератор може відхилити скаргу, видалити історію або коментар, якщо порушення підтверджене. Крім цього, йому доступний перегляд списку користувачів, пошук за іменем або email, а також блокування акаунтів на визначений строк чи без обмеження в часі. За потреби блокування можна скасувати. У таблиці 2.1 наведено стислий опис основних прецедентів системи.

Таблиця 2.1 – Основні прецеденти вебплатформи

Прецедент	**Актор**	**Опис**
Реєстрація	Гість	Створення облікового запису за email і паролем
Авторизація	Гість	Вхід до системи
Перегляд історій	Гість, читач	Ознайомлення зі списком опублікованих творів
Читання глави	Гість, читач	Перегляд тексту окремої глави
Пошук	Гість, читач	Пошук історій і авторів
Фільтрація та сортування	Гість, читач	Відбір творів за жанрами, статусом, кількістю глав та іншими параметрами
Створення історії	Автор	Додавання нового твору з основною інформацією

Продовження таблиці 2.1

Редагування історії	Автор	Зміна назви, опису, жанрів, статусу й обкладинки
Додавання глави	Автор	Створення нової частини історії
Редагування глави	Автор	Оновлення вже створеного тексту
Публікація матеріалу	Автор	Переведення історії або глави з чернетки у відкритий доступ
Коментування	Читач	Додавання коментарів і відповідей
Лайк історії	Читач	Позначення вподобаного твору
Додавання в закладки	Читач	Збереження історії для подальшого перегляду
Підписка на автора	Читач	Отримання сповіщень про нові публікації
Перегляд історії читання	Читач	Відображення прочитаних і переглянутих творів
Отримання рекомендацій	Читач	Персоналізована добірка історій на основі інтересів
Скарга на контент	Читач	Подання скарги на історію або коментар
Модерація скарг	Модератор	Перегляд і обробка звернень користувачів
Видалення порушень	Модератор	Видалення історій або коментарів після перевірки
Блокування користувачів	Модератор	Тимчасове або постійне обмеження доступу
Управління користувачами	Модератор	Перегляд списку акаунтів і пошук потрібного користувача

Побудована Use Case діаграма дозволила впорядкувати функції вебзастосунку та визначити межі доступу для кожної ролі. На її основі було простіше спроєктувати серверні маршрути, структуру сторінок і взаємодію між окремими модулями системи.

2.2 Архітектура системи

Під час розробки вебплатформи важливо заздалегідь визначити її загальну будову, розподіл відповідальності між окремими частинами та спосіб обміну даними. Саме архітектура задає основу для подальшої реалізації, впливає на зручність підтримки коду, можливість розширення функціоналу та стабільність роботи системи.

Для проєкту було обрано клієнт-серверну архітектуру з поділом на фронтенд і бекенд. Такий підхід добре підходить для вебзастосунків, у яких інтерфейс користувача і серверна логіка розвиваються окремо. Клієнтська частина відповідає за відображення сторінок, обробку дій користувача та надсилання запитів. Серверна частина приймає ці запити, виконує необхідну логіку, працює з базою даних і повертає результат у форматі JSON[20].

Загальну схему архітектури системи наведено на рис. 2.2. Користувач відкриває платформу в браузері, де завантажується застосунок, створений на React. Далі фронтенд звертається до REST API, реалізованого на Flask. Сервер обробляє запит, за потреби отримує або змінює дані в MySQL, після чого надсилає відповідь клієнту. Інтерфейс оновлюється без повного перезавантаження сторінки, що робить роботу з платформою комфортнішою.

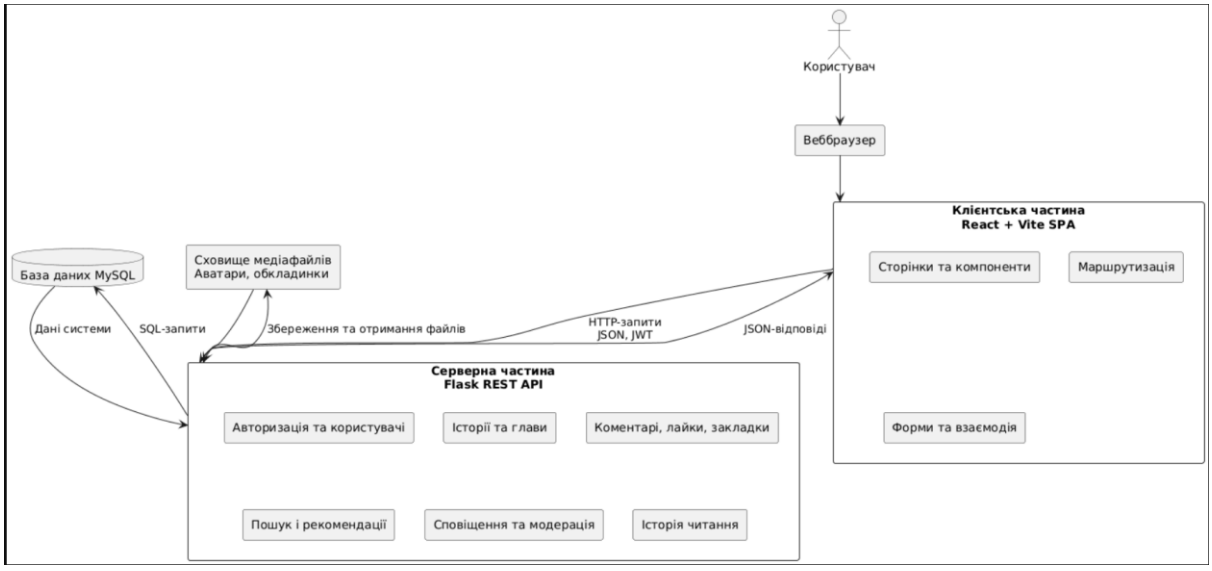


Рисунок 2.2 – Схема архітектури системи

Клієнтська частина побудована на компонентному підході. Інтерфейс складається з окремих багаторазових елементів, кожен із яких відповідає за свою ділянку функціоналу. Наприклад, компонент картки історії використовується на головній сторінці, у пошуку, у профілі користувача, у закладках і в добірках. Це дозволяє не дублювати однакову розмітку й спрощує внесення змін.

У структурі фронтенду можна виділити кілька основних частин. До першої належать сторінки, що відповідають окремим маршрутам: головна, сторінка історії, профіль, налаштування, історія читання, сторінка пошуку, панель модератора. Другу частину становлять компоненти інтерфейсу, зокрема банер, секції рекомендацій, форма коментарів, повідомлення, картки авторів та історій. Окремо винесено допоміжні функції й запити до API. Стилі організовані через SCSS-файли, прив’язані до відповідних сторінок і компонентів.

Для переходів між сторінками використовується react-router-dom. Завдяки цьому маршрутизація відбувається на стороні клієнта, а сторінка не перезавантажується повністю при кожному переході. Користувач отримує більш плавну навігацію, що особливо помітно під час роботи з профілем, каталогом історій або читанням окремих глав.

Бекенд реалізовано у вигляді REST API. Він складається з окремих маршрутів, що відповідають за конкретні групи завдань. Сервер не зберігає стан користувацької сесії між запитами. Для перевірки авторизації використовується JWT-токен, який надсилається у заголовку Authorization. Після перевірки токена сервер визначає користувача та його права доступу[20].

Серверна частина поділена на модулі за допомогою Flask Blueprint. Це дозволило розділити функціонал на логічні блоки:

- auth для реєстрації та входу;
- users для профілів, підписок і налаштувань;
- stories для історій, обкладинок і статистики;

- texts для роботи з главами;
- comments для коментарів і відповідей;
- likes та bookmarks для взаємодії з контентом;
- notifications для сповіщень;
- search для пошуку;
- history для історії читання;
- moderation для скарг і блокувань;
- recommendations для персоналізованих добірок.

Такий розподіл робить код зрозумілішим. Якщо потрібно змінити логіку коментарів або рекомендацій, достатньо працювати з відповідним модулем, не зачіпаючи решту системи.

Маршрути API побудовані відповідно до HTTP-методів. GET використовується для отримання даних, POST для створення нових записів, PUT для оновлення, DELETE для видалення. Це робить взаємодію між клієнтською та серверною частинами послідовною й зручною для подальшого тестування.

Робота з базою даних виконується через mysql-connector-python. У проєкті застосовуються параметризовані SQL-запити, що зменшує ризик SQL-ін’єкцій. Підключення до бази відкривається на час виконання конкретної операції та закривається після завершення роботи, що допомагає уникати зайвого навантаження на сервер.

Під час реалізації статистики для історій окрему увагу було приділено коректності підрахунків. Для визначення кількості переглядів, лайків і коментарів використовуються підзапити. Це рішення дало змогу уникнути помилкового дублювання даних, яке могло б виникнути при одночасному об’єднанні кількох таблиць через JOIN.

У системі також передбачено модуль рекомендацій. Він формує персоналізовану стрічку на основі активності користувача, зокрема лайків, закладок і прочитаних історій. Рекомендації обчислюються на сервері, а результати тимчасово зберігаються в окремій таблиці кешу. Це зменшує

кількість повторних обчислень і дозволяє швидше показувати добірки на головній сторінці.

Отже, обрана архітектура забезпечує чіткий поділ між інтерфейсом, серверною логікою та даними. Вона відповідає потребам розробленої платформи, дає змогу підтримувати код у впорядкованому вигляді та створює основу для подальшого розвитку функціоналу.

2.3 Проєктування бази даних

Проєктування бази даних є одним із найважливіших етапів розробки вебзастосунку. Саме на цьому рівні визначається, як будуть зберігатися користувачі, історії, глави, коментарі, закладки, рекомендації та інші дані, з якими працює система. Від структури бази залежить зручність подальшої розробки, коректність зв’язків між сутностями та швидкість виконання запитів.

Для вебплатформи було обрано реляційну базу даних MySQL. Такий підхід є доцільним, оскільки основні сутності системи мають чітку структуру й тісно пов’язані між собою. Наприклад, кожна історія належить конкретному автору, кожна глава пов’язана з певною історією, а коментар створюється до конкретної глави. Реляційна модель дає змогу описати такі зв’язки через зовнішні ключі та забезпечити цілісність даних.

Загальну ER-діаграму бази даних наведено на рис. 2.3. Вона відображає основні таблиці, ключові поля та зв’язки між ними.

Однією з головних таблиць є users. Вона містить дані зареєстрованих користувачів: унікальний ідентифікатор, ім’я, електронну пошту, хешований пароль, аватар, короткий опис профілю та додаткові службові поля. Також у цій таблиці зберігається роль користувача, ознака блокування, причина бану, дата завершення тимчасового обмеження та час останньої активності. Це

дозволяє реалізувати авторизацію, персональні профілі та модерацію акаунтів.

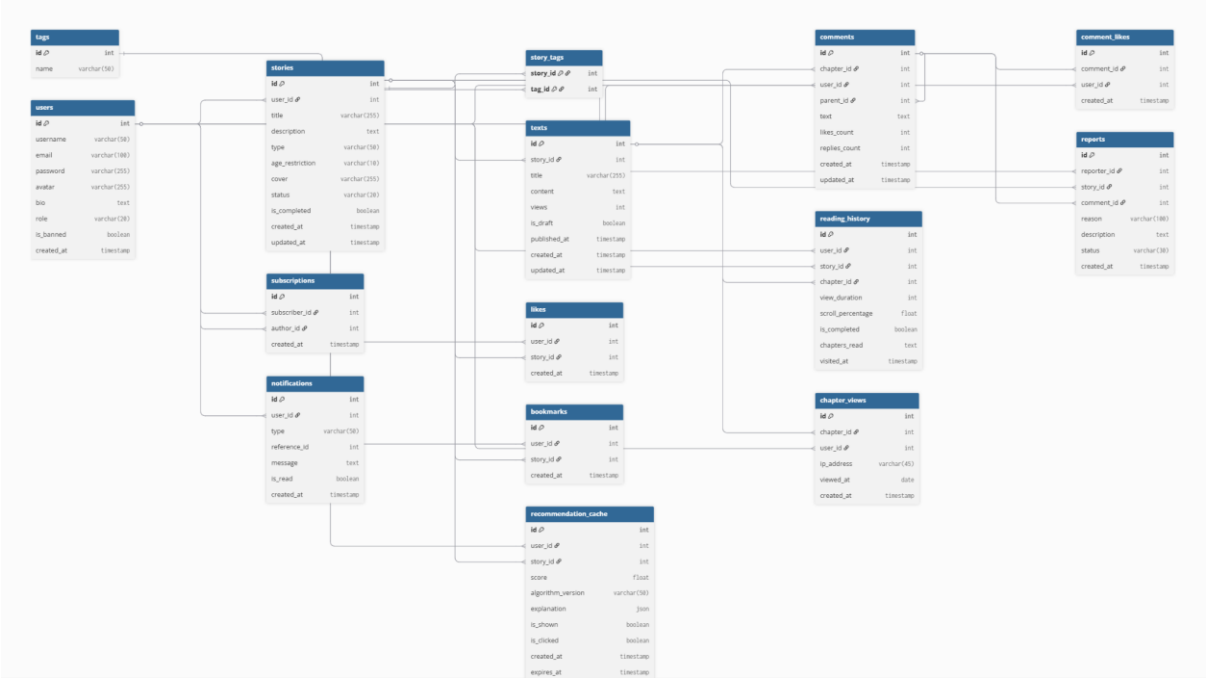


Рисунок 2.3 – ER-діаграма бази даних

Таблиця stories відповідає за збереження авторських історій. У ній містяться назва твору, опис, тип, вікове обмеження, статус публікації, інформація про завершеність історії, шлях до обкладинки, а також ідентифікатор автора. Через зовнішній ключ user_id кожна історія пов’язується з конкретним користувачем.

Окремі частини історій зберігаються у таблиці texts. Кожен запис відповідає одній главі та містить назву, HTML-вміст, належність до певної історії, ознаку чернетки, дату створення, дату публікації та кількість переглядів. Така структура дозволяє автору поступово публікувати твір і працювати з главами окремо.

Для жанрової класифікації використовується таблиця tags. Вона зберігає перелік доступних жанрів. Зв’язок між жанрами та історіями реалізовано через проміжну таблицю story_tags, оскільки одна історія може мати кілька жанрів, а один жанр може використовуватися в багатьох історіях.

Таблиці likes і bookmarks забезпечують взаємодію читачів із контентом. У likes зберігаються вподобання історій, а в bookmarks записи про додавання творів до закладок. У кожній з цих таблиць є зв’язок між користувачем та історією. Для уникнення дублювання передбачено унікальність пари user_id і story_id.

Коментування реалізовано через таблицю comments. Вона містить текст коментаря, автора, главу, до якої він належить, а також дату створення й редагування. Поле parent_id дозволяє організувати відповіді на коментарі. Якщо це поле порожнє, коментар є основним, а якщо містить значення, запис вважається відповіддю. Для швидкого відображення даних також зберігаються лічильники лайків і відповідей. Вподобання коментарів винесено в окрему таблицю comment_likes.

Таблиця subscriptions використовується для підписок на авторів. Вона фіксує, який користувач підписався на якого автора. Завдяки цьому система може показувати список підписок і створювати сповіщення про нові глави.

Для надсилання системних повідомлень використовується таблиця notifications. У ній зберігаються ідентифікатор отримувача, тип сповіщення, пов’язаний об’єкт, текст повідомлення, ознака прочитання та дата створення. Сповіщення формуються, наприклад, після нового коментаря, відповіді, лайка, закладки, підписки або видалення контенту модератором.

Таблиця reports потрібна для роботи системи скарг. Користувач може поскаржитися на історію або коментар, вказати причину та за бажанням додати пояснення. Поле статусу дозволяє відстежувати, чи очікує скарга розгляду, чи вже була вирішена або відхилена модератором.

Для збереження прогресу читання створено таблицю reading_history. Вона містить дані про користувача, історію, останню відкриту главу, час читання, максимальний відсоток прокрутки, статус завершення та перелік уже прочитаних глав. Завдяки цьому система може показувати історію переглядів, обчислювати прогрес і повертати користувача до матеріалу, який він читав раніше.

Окрему роль відіграє таблиця chapter_views. Вона потрібна для точнішого підрахунку переглядів глав. Перегляд фіксується із прив’язкою до користувача або IP-адреси та дати. Це дозволяє не збільшувати лічильник багаторазово при повторному відкритті тієї самої глави протягом доби.

Для роботи персоналізованої рекомендаційної системи використовується таблиця recommendation_cache. У ній зберігаються вже обчислені рекомендації для певного користувача: ідентифікатор історії, числова оцінка релевантності, версія алгоритму, ознаки показу та переходу, дата створення й час завершення дії кешу. У поточній реалізації для рекомендацій використовується алгоритм на основі машинного навчання, а кешування дозволяє не виконувати однакові розрахунки при кожному відкритті головної сторінки.

Зв’язки між таблицями організовано через зовнішні ключі. Це дає змогу підтримувати цілісність даних, наприклад не створювати главу без історії або коментар без автора. Для полів, що часто використовуються у фільтрації та пошуку, доцільно застосовувати індексацію. У межах розробленої системи особливо важливими є індекси за ідентифікаторами користувачів, історій, глав, датами створення та полями, які беруть участь у зв’язках між таблицями.

Під час розробки запитів також враховувалася точність підрахунку статистики. Для кількості переглядів, лайків і коментарів у низці запитів застосовуються вкладені підзапити. Це дозволяє уникнути дублювання рядків, яке може виникнути при одночасному об’єднанні кількох таблиць.

Отже, спроєктована база даних охоплює весь основний функціонал вебплатформи: користувачів, авторські тексти, взаємодію з контентом, модерацію, прогрес читання, сповіщення та персоналізовані рекомендації. Її структура відповідає логіці застосунку й забезпечує основу для стабільної роботи системи.

2.4 Розробка серверної частини

Серверну частину вебплатформи реалізовано мовою Python із використанням фреймворку Flask. У межах проєкту цей інструмент виявився зручним для побудови логіки, що обробляє запити від клієнтської частини, працює з базою даних, перевіряє права користувачів і повертає результати у форматі JSON. Оскільки загальні принципи архітектури вже були розглянуті в попередньому підрозділі, далі увагу зосереджено саме на тому, як реалізовано основні серверні можливості вебзастосунку.

Центральним файлом серверної частини є app.py. У ньому підключаються всі функціональні модулі, налаштовуються JWT-токени, CORS, обробники типових помилок авторизації та проміжна перевірка користувача перед виконанням запитів. Окремо реалізовано логіку, яка перевіряє, чи не заблокований акаунт, і за потреби забороняє доступ до приватних дій. Також під час активності користувача оновлюється поле останньої активності, що дозволяє зберігати актуальні службові дані в профілі.

Серверний код поділено на окремі модулі відповідно до функціональних частин системи. Модуль auth відповідає за реєстрацію та авторизацію, users працює з профілями, підписками й налаштуваннями, stories обробляє історії, а texts відповідає за глави. Окремо винесено коментарі, вподобання, закладки, сповіщення, історію читання, пошук, модерацію та рекомендації. Така організація не дублює опис архітектури, але важлива з практичного погляду, оскільки саме завдяки цьому серверний код залишається зрозумілим і придатним для подальшого розширення.

Реєстрація користувача передбачає створення акаунта з псевдонімом, електронною поштою та паролем. Перед збереженням пароля до бази даних він перетворюється на хеш за допомогою бібліотеки Flask-Bcrypt. Завдяки цьому початкове значення пароля не зберігається у відкритому вигляді. Під час входу сервер порівнює введений пароль із хешем і, якщо дані коректні,

формує JWT-токен. Токен використовується для підтвердження авторизації в подальших запитах і передається в заголовку Authorization. У застосунку строк дії токена встановлено на 14 днів. Якщо він відсутній, пошкоджений або вже неактуальний, сервер повертає відповідне повідомлення про помилку[16].

Для маршрутів, що потребують авторизації, застосовується декоратор @jwt_required(). Він використовується під час створення історій, редагування профілю, роботи із закладками, додавання коментарів, перегляду власної історії читання та інших операцій, які не повинні бути доступні анонімним відвідувачам. Окремо в серверній частині передбачено перевірку прав модератора або адміністратора. Такий механізм використовується в маршрутах панелі модерації, де звичайний користувач не повинен мати доступу до перегляду скарг, блокування акаунтів або видалення матеріалів [13].

Особливу увагу приділено роботі з базою даних. Усі SQL-запити виконуються через параметризований синтаксис. Це означає, що значення користувача передаються не як частина SQL-рядка, а окремими параметрами. Такий підхід знижує ризик SQL-ін’єкцій і робить взаємодію з базою більш безпечною. З’єднання з MySQL відкривається лише на час виконання конкретної операції та закривається після завершення запиту, що допомагає уникати непотрібного навантаження на сервер.

Велика частина серверної логіки пов’язана зі створенням і редагуванням авторських історій. Після авторизації користувач може додати новий твір, указавши назву, опис, тип, вікове обмеження та вибрані жанри. На цьому етапі історія зберігається як чернетка, після чого автор може доповнювати її главами й пізніше публікувати. Передбачено можливість редагувати основну інформацію про твір, змінювати його статус, позначати історію як завершену, а також оновлювати склад жанрів.

Для кожної історії можна завантажити обкладинку. Сервер перевіряє допустимий формат файлу, обробляє зображення й зберігає його в директорії

завантажень. За потреби автор може замінити або видалити вже наявну обкладинку. Це реалізовано окремими маршрутами, що дозволяє клієнтській частині відправляти файли незалежно від основної форми редагування історії.

Глави створюються та редагуються окремо від загальної інформації про історію. Коли автор працює в редакторі, сервер отримує назву глави, HTML-вміст і статус публікації. Текст можна зберегти як чернетку або одразу зробити доступним читачам. Якщо автор редагує вже створену главу, бекенд перевіряє право власності й лише після цього дозволяє оновлення. Такий підхід не дає користувачу змінювати чужі тексти через прямий запит до API.

Під час перегляду історії сервер повертає не лише її основні поля, а й допоміжні дані, потрібні для інтерфейсу. До них належать кількість глав, загальні перегляди, кількість лайків і коментарів. Для цих показників використовуються окремі підзапити, оскільки звичайне об’єднання кількох таблиць у складному JOIN може призвести до дублювання рядків і неточного підрахунку статистики. Наприклад, при одночасному зв’язуванні глав, лайків і коментарів одна й та сама історія може повторюватися в результаті кілька разів. Використання підзапитів дозволяє цього уникнути й отримувати коректні значення.

Для читачів реалізовано повний набір базових дій із контентом. Користувач може поставити лайк історії, зняти його, додати твір до закладок або прибрати його звідти. На рівні бази даних і серверної логіки передбачено перевірки, які не дозволяють створювати дублікати таких записів. Коментарі додаються до конкретної глави, а відповіді на них зберігаються через зв’язок із батьківським коментарем. Завдяки цьому можна вести обговорення не лише окремої історії, а саме певного її фрагмента.

Окремо реалізовано лайки для коментарів. Сервер зберігає факт вподобання та повертає оновлену кількість реакцій, щоб інтерфейс міг одразу показати зміни без перезавантаження сторінки. Також передбачено

редагування й видалення власних коментарів. Перед виконанням таких дій сервер перевіряє, чи справді поточний користувач є автором повідомлення.

Підписки на авторів працюють як механізм підтримки постійної взаємодії між читачем і творцем контенту. Після оформлення підписки користувач може бачити автора у власному списку підписок, а сам автор отримує відповідне сповіщення. Крім того, підписники повідомляються про появу нових глав, що допомагає утримувати інтерес до продовжуваних історій.

Система сповіщень реалізована через окремий серверний модуль. Повідомлення створюються для подій, які мають значення для користувача: нова підписка, новий коментар, відповідь на власний коментар, вподобання історії, поява нової глави, видалення контенту модератором. Кожне сповіщення записується до бази даних із типом події, текстом повідомлення, ідентифікатором пов’язаного об’єкта та ознакою прочитання. Завдяки цьому в інтерфейсі можна показувати лічильник непрочитаних повідомлень і окремий список останніх подій.

Для зручного пошуку матеріалів створено серверний модуль, що обробляє текстовий запит і повертає збіги серед історій та авторів. Пошук враховує назву твору, опис і псевдонім автора. У відповіді також передається допоміжна статистика для відображення карток історій, тому результати можна одразу виводити у тому самому форматі, що й на інших сторінках вебзастосунку. Пошук реалізований і на окремій сторінці з результатами, і у вигляді швидкого випадного списку в шапці сайту.

Окремим серверним модулем реалізовано персоналізовані рекомендації. Вони формуються для зареєстрованих користувачів із урахуванням уже наявної активності в системі. Алгоритм аналізує історії, з якими користувач раніше взаємодіяв, і зіставляє їх із доступними творами за текстовими характеристиками. На основі цього формується добірка матеріалів, які можуть відповідати його інтересам. Таким чином,

рекомендації не є випадковим списком популярних історій, а залежать від поведінки конкретного читача.

Для зменшення навантаження на сервер результати рекомендацій зберігаються в таблиці recommendation_cache. У ній містяться ідентифікатор користувача, рекомендована історія, числова оцінка відповідності, версія алгоритму, ознаки показу й переходу, а також строк актуальності запису. Поки кеш не прострочено, система використовує вже готовий список. Після завершення терміну дії добірка може бути сформована повторно. Такий підхід дає змогу не виконувати однакові обчислення при кожному відкритті головної сторінки.

Модуль модерації призначений для роботи зі скаргами та проблемним контентом. Користувач може поскаржитися на історію або окремий коментар, указавши причину звернення. Після цього запис потрапляє до відповідної таблиці й стає доступним у панелі модератора. Модератор може переглянути скаргу, оцінити її обґрунтованість, відхилити звернення або видалити порушення. Також йому доступний список користувачів із пошуком за іменем або електронною поштою. За потреби модератор може тимчасово або постійно заблокувати акаунт. Якщо блокування є тимчасовим, у базі зберігається дата його завершення.

При видаленні контенту або застосуванні обмежень користувач отримує повідомлення. Це важливо, оскільки дозволяє зробити дії модератора зрозумілими з боку автора, а не залишати видалення матеріалу без пояснення. У майбутньому такий підхід також можна розвинути через окрему історію модераторських рішень.

Ще одним важливим елементом серверної частини є відстеження прогресу читання. Під час перебування користувача на сторінці глави клієнтська частина періодично надсилає на сервер дані про тривалість перегляду й відсоток прокрутки сторінки. Бекенд оновлює відповідний запис у таблиці reading_history, фіксуючи історію, главу, час читання та ознаку

завершення. Якщо користувач відкриває той самий твір пізніше, система може показати, що він уже читав його раніше, а також відобразити прогрес.

Поряд із автоматичним відстеженням передбачено ручне позначення глави як прочитаної. Це корисно в тих випадках, коли користувач дочитав матеріал, але за технічними умовами система не встигла зафіксувати достатній прогрес. Дані про історію переглядів використовуються не лише у відповідній сторінці профілю, а й можуть враховуватися під час формування персональних рекомендацій.

Для сумісності з клієнтською частиною, яка під час розробки працює на іншому локальному порту, налаштовано CORS. Завдяки цьому браузер може коректно надсилати запити між фронтендом і бекендом. Обробка помилок JWT також винесена в окремі функції. Для відсутнього, недійсного та простроченого токена сервер повертає структуровану JSON-відповідь, що дає змогу фронтенду передбачувано реагувати на такі ситуації. Основні API-ендпоїнти серверної частини наведено у таблиці 2.2.

Таблиця 2.2 – Основні API-ендпоїнти серверної частини

Метод	**URL**	**Опис**
POST	/api/register	Реєстрація нового користувача
POST	/api/login	Авторизація та отримання токена
GET	/api/users/me	Дані поточного користувача
GET	/api/stories/all	Список опублікованих історій
GET	/api/stories/popular	Популярні історії
POST	/api/stories/add	Створення нової історії
PUT	/api/stories/:id	Оновлення історії
DELETE	/api/stories/:id	Видалення історії
POST	/api/stories/:id/upload-cover	Завантаження обкладинки
GET	/api/texts/:id	Отримання глави
POST	/api/texts/chapter	Створення нової глави

Продовження таблиці 2.2

PUT	/api/texts/chapter/:id	Редагування глави
GET	/api/comments/chapter/:id	Коментарі до глави
POST	/api/comments/	Додавання коментаря
POST	/api/likes/story/:id	Лайк історії
POST	/api/bookmarks/:id	Додавання історії в закладки
GET	/api/search	Пошук історій та авторів
GET	/api/recommendations/feed	Персональні рекомендації
GET	/api/history/	Історія переглядів
POST	/api/history/track	Збереження прогресу читання
GET	/api/notifications/	Перегляд сповіщень
GET	/api/moderation/reports	Отримання скарг
POST	/api/moderation/users/:id/ban	Блокування користувача

Таким чином, серверна частина забезпечує не лише базові операції зі створення, редагування та видалення даних, а й усю прикладну логіку платформи. Вона підтримує роботу авторів і читачів, контролює доступ до приватних дій, обробляє активність користувачів, формує сповіщення, рекомендації та надає інструменти для модерації контенту. У результаті бекенд виступає повноцінною основою.

2.5 Розробка клієнтської частини

Клієнтську частину вебзастосунку створено з використанням бібліотеки React та інструмента збірки Vite. React було обрано через зручний компонентний підхід, який добре підходить для розробки інтерфейсів із великою кількістю повторюваних елементів. У межах проєкту це особливо корисно, оскільки ті самі компоненти використовуються на різних сторінках,

але зберігають однакову логіку та зовнішній вигляд. Vite застосовано для запуску проєкту в режимі розробки та швидкого оновлення змін у коді.

Інтерфейс працює як односторінковий застосунок. Перехід між сторінками виконується без повного перезавантаження браузера, що робить навігацію швидшою та зручнішою для користувача. Для маршрутизації використовується react-router-dom. Верхня панель навігації та нижній колонтитул залишаються спільними елементами для більшості сторінок, а центральна частина змінюється залежно від обраного маршруту.

Структура фронтенду організована за функціональним принципом. Окремі сторінки відповідають за самостійні сценарії роботи користувача: головна сторінка, каталог історій, сторінка окремого твору, перегляд глави, пошук, особистий профіль, публічний профіль автора, сторінка створення історії, текстовий редактор, історія переглядів, панель модератора, налаштування профілю, сторінки входу та реєстрації. Кожна з них складається з менших компонентів, що дозволяє не дублювати код і зберігати однакову логіку відображення в різних частинах системи.

Одним із найбільш уживаних компонентів є StoryCard. Він відповідає за відображення короткої інформації про історію та використовується на головній сторінці, у пошуку, у списках популярних і нових творів, у профілі користувача, на сторінці закладок і вподобаних історій. Компонент показує назву твору, автора, опис, жанри, обкладинку, вікове обмеження, статус історії та основну статистику. Якщо обкладинка відсутня, відображається стилізований текстовий плейсхолдер із назвою та автором. Для компактнішого вигляду числові показники форматуються, наприклад тисячі подаються у вигляді скорочених значень.

На головній сторінці застосовується компонент StorySection, який формує горизонтальні добірки історій. Кожна секція завантажує власний набір даних із сервера та відображає його у вигляді прокручуваного списку карток. У проєкті передбачено секції персональних рекомендацій, добірки дня, менш помітних історій для відкриття нового контенту, популярних

творів, нових публікацій, нещодавно оновлених історій, творів із великою кількістю переглядів і завершених історій. Горизонтальна прокрутка доповнена кнопками зі стрілками, які з’являються лише тоді, коли список виходить за межі видимої області. Приклад головної сторінки вебплатформи з добірками історій наведено на рисунку 2.4.

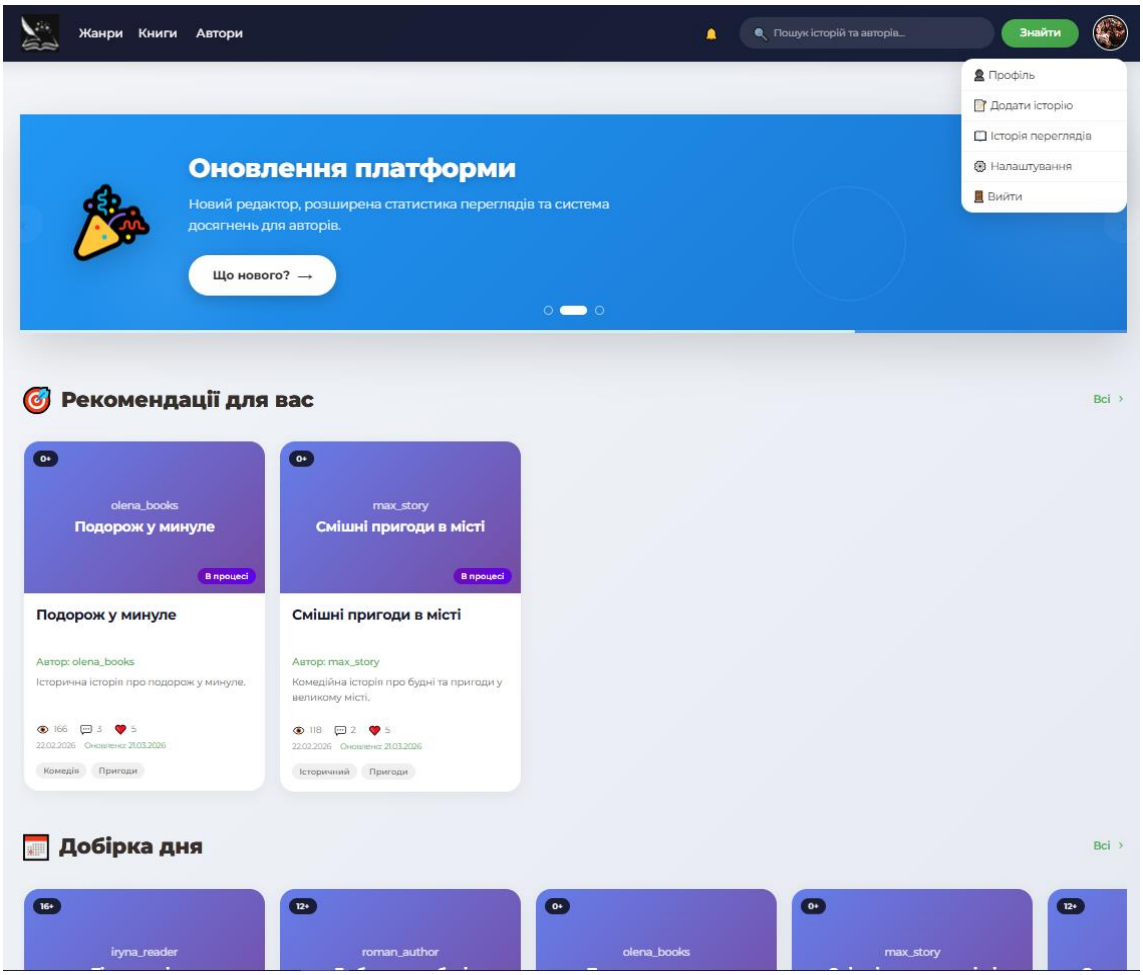


Рисунок 2.4 – Головна сторінка вебплатформи з добірками історій

Окреме місце в інтерфейсі займає компонент сповіщень, розміщений у верхній панелі. Він показує кількість непрочитаних повідомлень і дає змогу відкрити випадний список останніх подій. Користувач може переглянути нові сповіщення, перейти до пов’язаного об’єкта або позначити повідомлення як прочитані. Для цього компонент звертається до серверних маршрутів отримання списку сповіщень, лічильника непрочитаних записів і

їх оновлення. Такий підхід дозволяє користувачу швидко помічати нові коментарі, відповіді, підписки та оновлення авторів, за якими він стежить.

Сторінка окремої історії реалізована так, щоб читач бачив усю важливу інформацію без зайвих переходів. На ній відображаються назва, автор, жанри, вікове обмеження, обкладинка, короткий опис, статус публікації, позначка завершеності, список глав і статистика. Для читача передбачено кнопки вподобання, додавання до закладок і подання скарги на невідповідний контент. Якщо сторінку переглядає автор цієї історії, стають доступними додаткові дії: редагування опису, жанрів і статусу, завантаження або видалення обкладинки, створення нової глави, редагування й видалення вже наявних глав. Частина цих операцій виконується через модальні вікна, щоб не переносити користувача на іншу сторінку без потреби. Приклад сторінки перегляду інформації про історію наведено на рисунку 2.5.

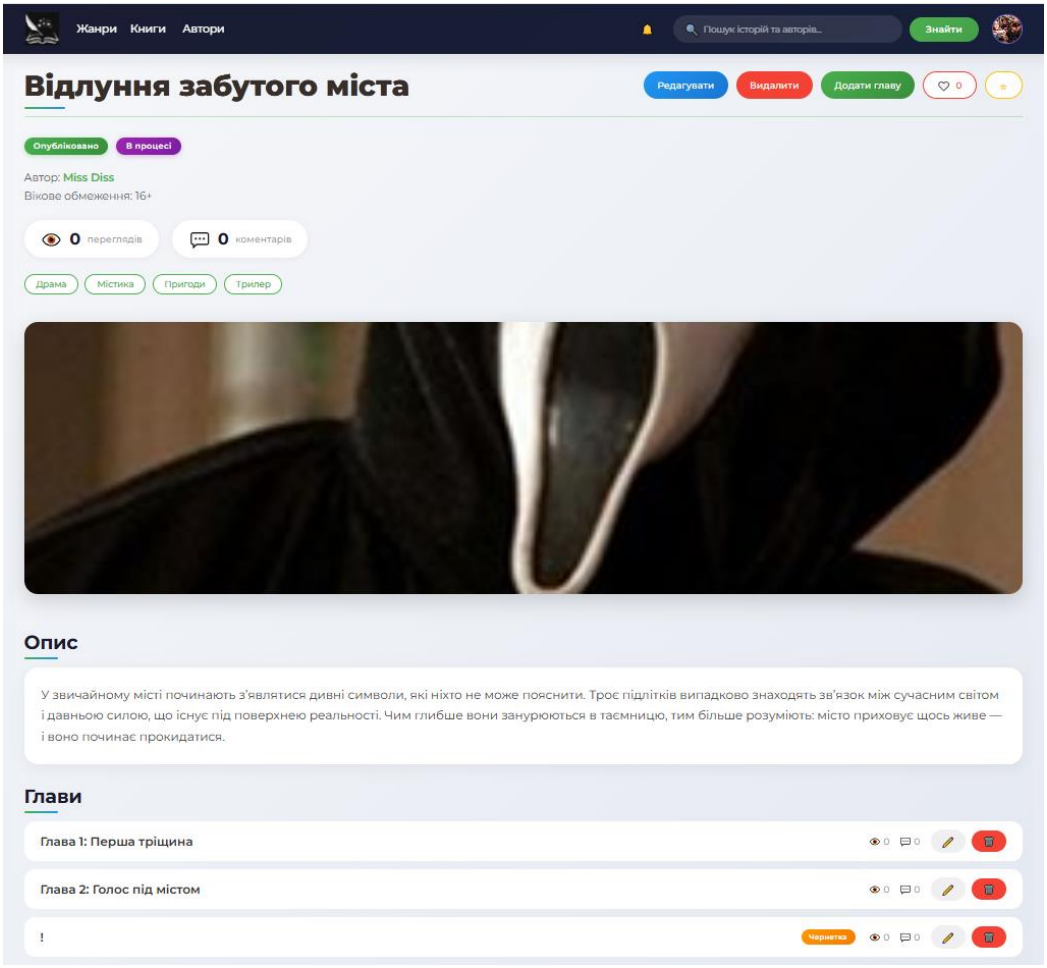


Рисунок 2.5 – Сторінка перегляду інформації про історію

Для створення історій реалізовано окрему форму. Автор вводить назву, опис, обирає тип твору, вікове обмеження та жанри через мультивибір. Також можна одразу завантажити обкладинку. На стороні клієнта перевіряється довжина назви й опису, наявність принаймні одного жанру, допустимий тип файлу та його розмір. Після успішного створення історії користувач переходить на її сторінку та може продовжити роботу з главами. Форму створення нової історії наведено на рисунку 2.6.

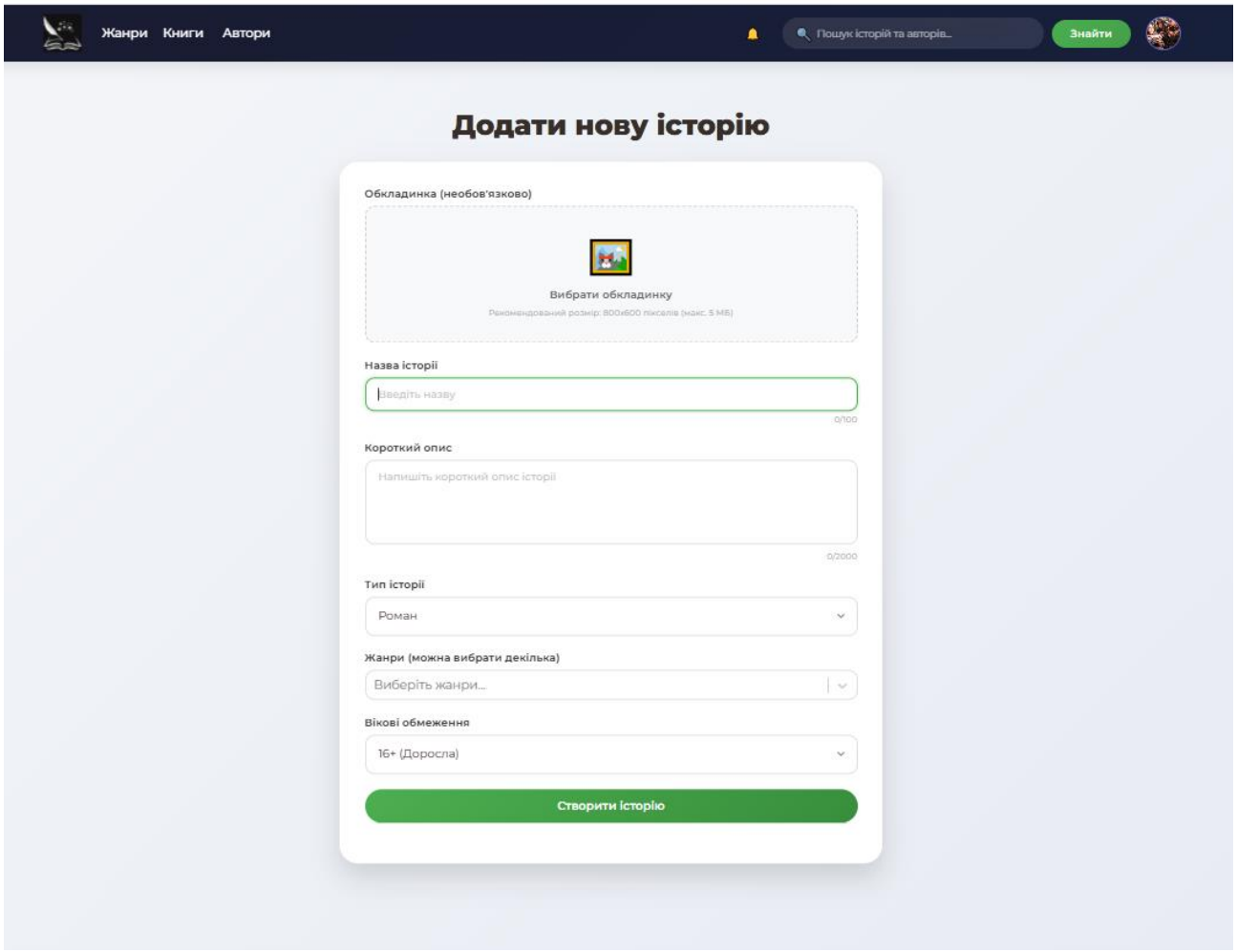


Рисунок 2.6 – Форма створення нової історії

Редактор глав побудовано на базі Tiptap. Він підтримує базове форматування тексту, зокрема жирний і курсивний шрифт, підкреслення, закреслення, вирівнювання, а також скасування й повторення дій. У редакторі можна створити нову главу або відкрити вже наявну для

редагування. Перед збереженням перевіряється, чи заповнено назву та вміст. Для нових неопублікованих матеріалів передбачено локальне збереження чернетки, щоб випадкове оновлення сторінки не призвело до втрати набраного тексту. Під час редагування вже створеної глави також працює автоматичне періодичне збереження чернетки. Інтерфейс редактора для створення та редагування глави наведено на рисунку 2.7.

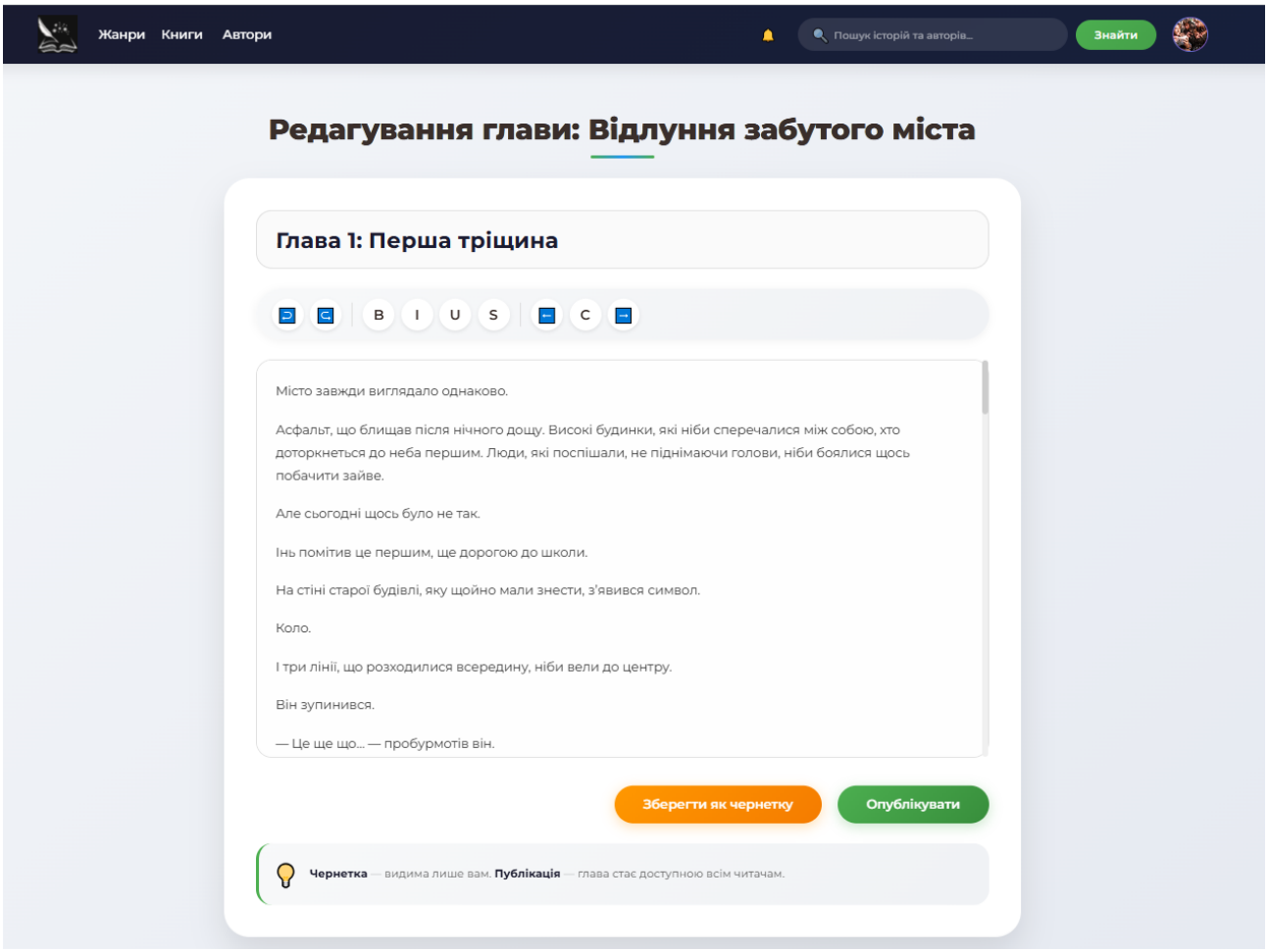


Рисунок 2.7 – Інтерфейс редактора для створення та редагування глави

Сторінка читання глави має спокійніший вигляд, ніж сторінки каталогу чи профілю. У центрі уваги знаходиться сам текст. У верхній частині розміщено повернення до сторінки історії, назву глави та автора. Нижче знаходиться вміст тексту, а наприкінці сторінки може відображатися кнопка переходу до наступної глави. Якщо користувач авторизований, система відстежує прогрес читання: періодично передається час перебування на

сторінці та відсоток прокрутки. Також читач може вручну позначити главу як прочитану. Сторінку читання глави з блоком коментарів наведено на рис. 2.8.

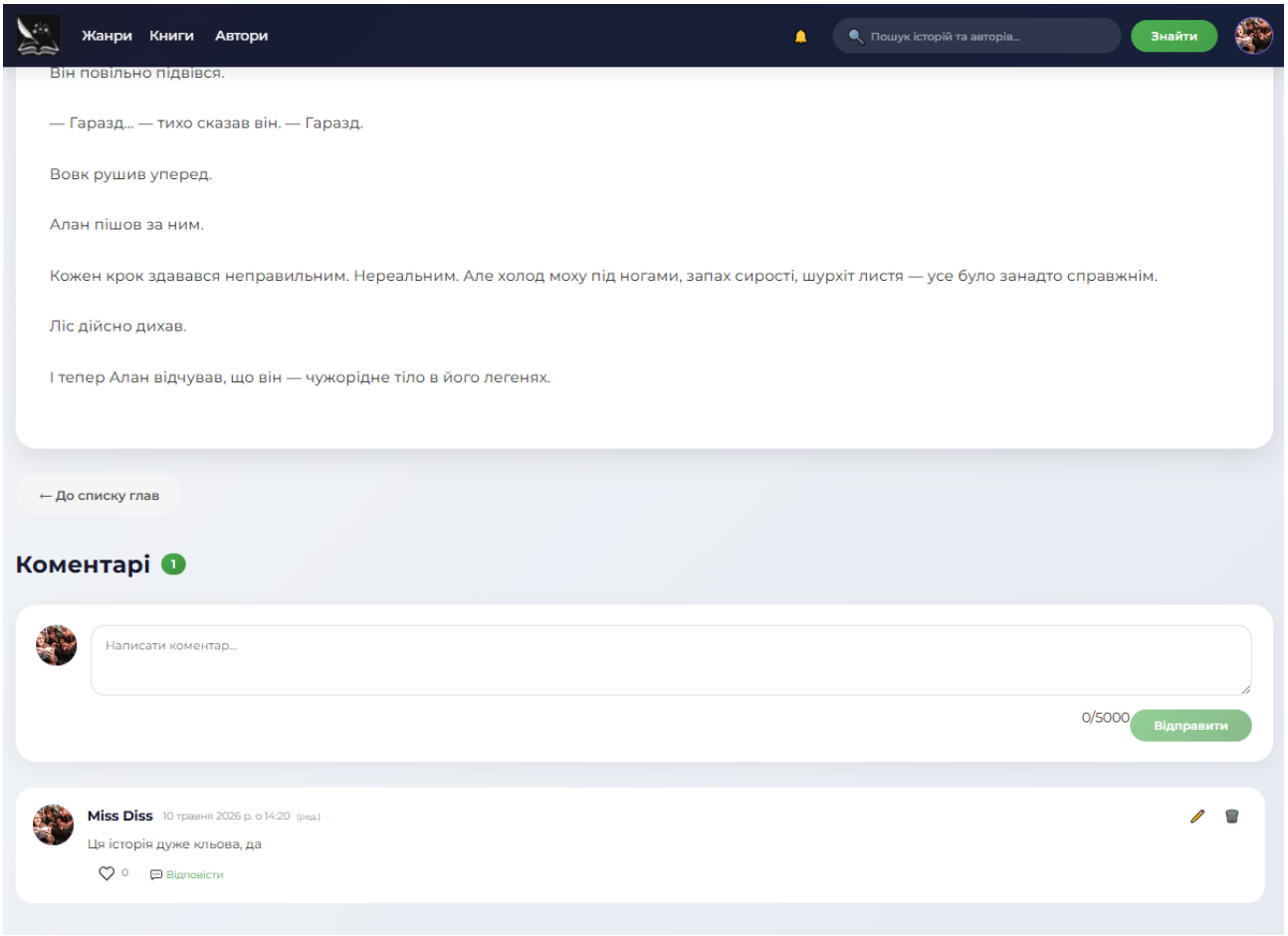


Рисунок 2.8 – Сторінка читання глави з блоком коментарів

Під главою розташовано блок коментарів. Він дозволяє авторизованим користувачам залишати нові повідомлення, відповідати іншим читачам, ставити лайки коментарям, редагувати або видаляти власні записи. Якщо користувач не ввійшов у систему, замість форми він бачить запрошення авторизуватися або створити акаунт. Для великої кількості коментарів реалізовано поступове підвантаження наступних записів під час прокрутки сторінки, що дає змогу не завантажувати весь список одразу.

Сторінка пошуку виконує дві ролі. По-перше, вона показує результати загального текстового запиту серед історій і авторів. По-друге, дозволяє окремо переглядати знайдені твори та профілі авторів через вкладки. Для

користувачів, які ще не ввели запит, виводяться популярні варіанти пошуку. Окремо в каталозі історій реалізовано розширену систему фільтрів. Там можна обирати жанри для включення або виключення, задавати тип твору, вікове обмеження, статус завершеності, кількість глав, мінімальні перегляди чи лайки, дати публікації та спосіб сортування. Активні умови відображаються у вигляді окремих позначок, які можна прибирати по одній або скинути всі одразу.

Особистий кабінет користувача поєднує інформаційний блок і добірки пов’язаного контенту. У верхній частині розміщені аватар, ім’я, електронна пошта, дата реєстрації та статистика активності. Нижче доступні вкладки з власними історіями, уподобаними творами й закладками. Також користувач може переглянути списки підписників і підписок у модальних вікнах. Окремо реалізовано сторінку налаштувань, де дозволено змінювати профільні дані, завантажувати новий аватар і оновлювати пароль.

Публічний профіль автора містить основну інформацію про користувача, кількість підписників, статистику публікацій і перелік його опублікованих історій. Якщо відвідувач авторизований і переглядає чужий профіль, він може підписатися на автора або скасувати підписку. Також доступний перегляд списку підписників.

Сторінка авторів виводить список користувачів, які мають опубліковані історії. Картка автора містить аватар, ім’я, кількість творів, перегляди, лайки, популярні жанри та дату приєднання до платформи. Передбачено пошук за іменем, сортування за популярністю, кількістю історій, переглядами та датою реєстрації, а також пагінацію, щоб список залишався зручним навіть при збільшенні кількості користувачів.

Історія переглядів відображає твори, які користувач уже відкривав. Для кожного запису показується обкладинка, автор, прогрес читання, кількість прочитаних глав, час взаємодії та дата останнього перегляду. Користувач може видалити один або кілька записів, а також повністю очистити історію. Для групових дій передбачено режим вибору.

Окрема сторінка призначена для модератора. У ній реалізовано перегляд скарг на історії та коментарі, ухвалення рішення щодо кожного звернення, видалення проблемного контенту, пошук користувачів і керування блокуваннями. Таблиці й картки в цьому розділі дають змогу швидко переглянути причину скарги, автора повідомлення, фрагмент проблемного контенту та поточний статус розгляду. Панель модератора для перегляду скарг і керування користувачами наведено на рисунку 2.9.

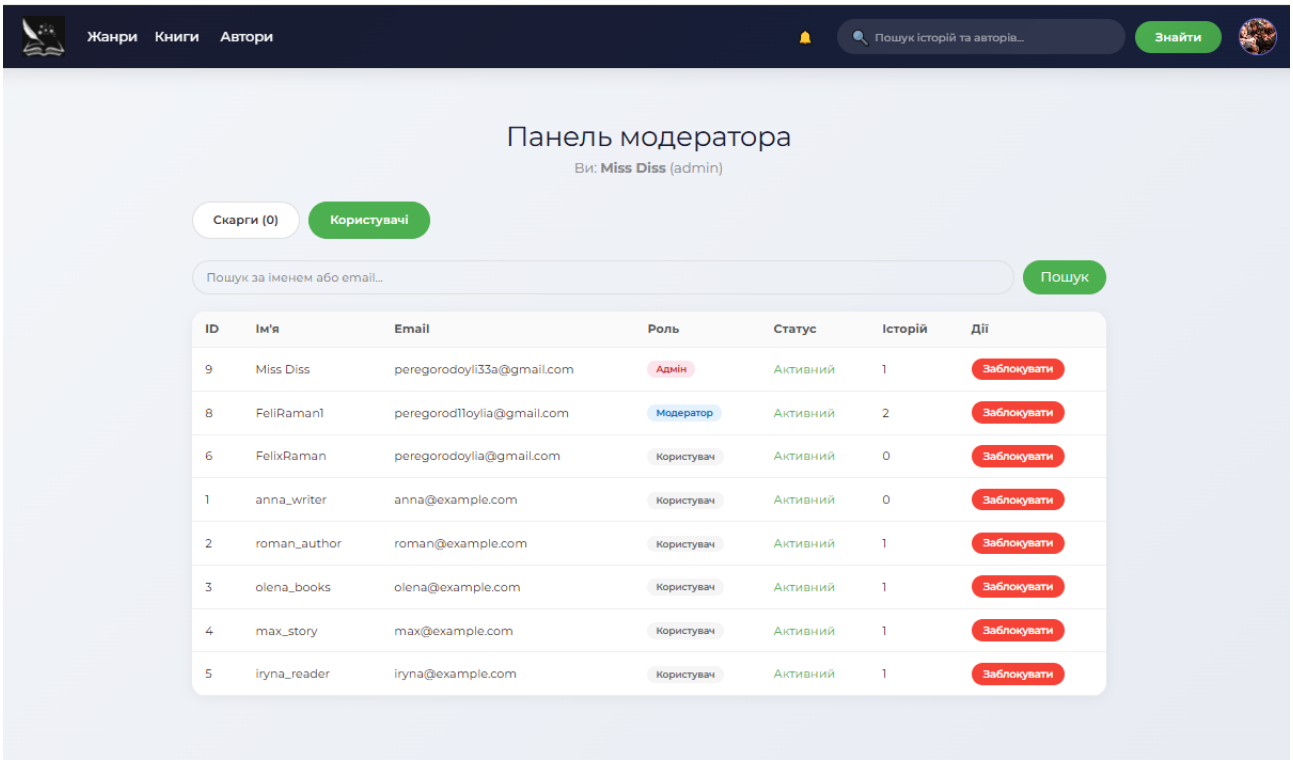


Рисунок 2.9 – Панель модератора для перегляду скарг і керування користувачами

У роботі з формами й інтерфейсними елементами використано кілька допоміжних бібліотек. React-Select застосовується для вибору жанрів, Tiptap для редактора тексту, date-fns для форматування дат на сторінці історії переглядів. У більшості компонентів запити надсилаються через fetch, а в модулі коментарів використовується axios, оскільки він зручний для роботи з послідовними запитами, оновленнями стану та обробкою помилок.

Стилізацію реалізовано через SCSS. Для сторінок і компонентів створено окремі файли стилів, а глобальні правила винесено в загальні SCSS-файли. Це дозволяє зберігати оформлення впорядкованим і не змішувати стилі різних частин інтерфейсу. Візуально вебзастосунок побудований у єдиному стилі з картками контенту, акцентними кнопками, м’якими тінями та зрозумілою ієрархією блоків. Для різних розмірів екрана передбачено адаптивне відображення, що дає змогу користуватися платформою і з комп’ютера, і з мобільного пристрою.

Отже, клієнтська частина забезпечує повний набір інтерфейсів для читачів, авторів і модераторів. Вона охоплює створення та редагування текстів, перегляд історій, взаємодію з контентом, персональні добірки, профілі, історію читання, пошук і модерацію. Компонентний підхід у React дозволив зробити структуру інтерфейсу послідовною, а окремі сторінки достатньо гнучкими для подальшого розвитку платформи.

2.6 Алгоритм рекомендацій

Однією з важливих частин розробленої вебплатформи є система персоналізованих рекомендацій. Її призначення полягає в тому, щоб допомогти користувачу швидше знаходити історії, які можуть відповідати його інтересам. Для платформи з авторськими текстами це особливо корисно, оскільки зі збільшенням кількості публікацій звичайного перегляду каталогу або ручного пошуку може бути недостатньо. Рекомендації дають змогу показати читачеві не випадковий перелік творів, а добірку, сформовану з урахуванням його попередньої активності.

У поточній реалізації рекомендаційна система побудована на основі методів машинного навчання для аналізу текстових характеристик історій. Основою алгоритму є TF-IDF-векторизація та обчислення косинусної схожості. Такий підхід дозволяє представити кожну історію у вигляді

числового вектора, а потім порівнювати ці вектори між собою. Для формування текстового представлення твору використовуються його назва, опис, тип і жанри. Саме ці дані найкраще відображають загальний зміст історії та її тематичну спрямованість [17].

Схема формування персоналізованих рекомендацій наведена на рис. 2.10.

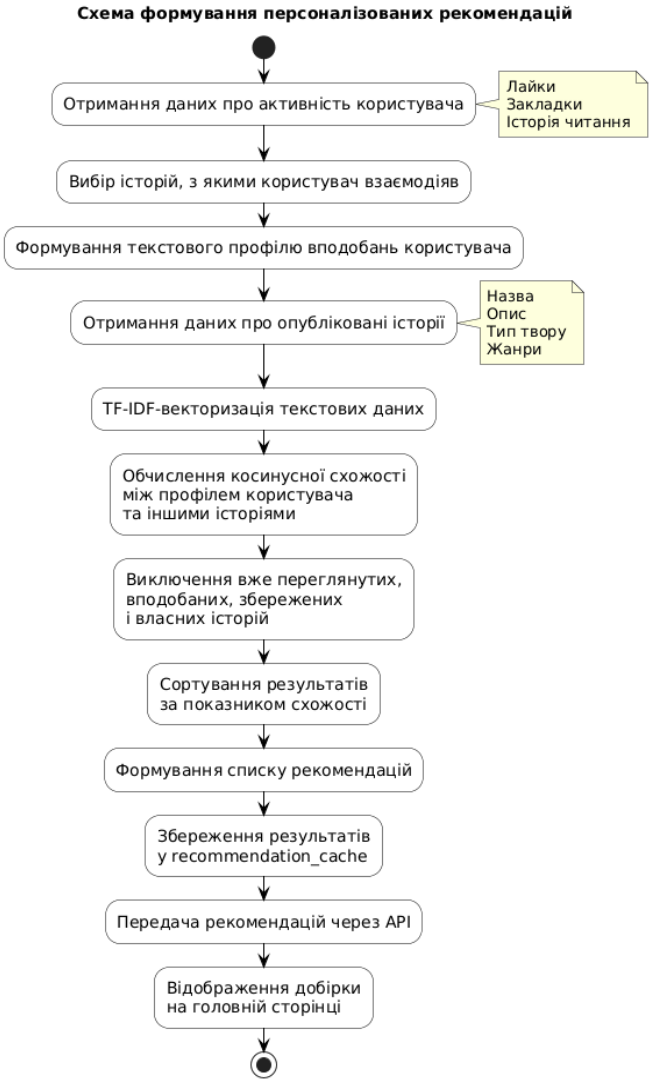


Рисунок 2.10 – Схема формування персоналізованих рекомендацій

Спочатку система отримує всі опубліковані історії, доступні для рекомендацій. Для кожної з них формується суцільний текстовий рядок, що складається з основних описових полів. Після цього застосовується

TfidfVectorizer, який перетворює тексти на матрицю ознак. У налаштуваннях використано обмеження до 5000 ознак, а також аналіз не лише окремих слів, а й пар слів. Це допомагає враховувати короткі смислові поєднання, які можуть бути важливими для опису жанру або тематики твору.

Персоналізація рекомендацій відбувається через побудову профілю інтересів користувача. Система аналізує історії, з якими він уже взаємодіяв: поставив лайк, додав у закладки, дочитав повністю або відкривав раніше. Різні типи дій мають різну вагу. Найсильнішим сигналом вважається лайк, дещо меншу вагу має закладка, далі враховується завершене та часткове читання. Це зроблено для того, щоб система краще розрізняла випадковий перегляд і справжню зацікавленість користувача.

На основі цих взаємодій формується узагальнений вектор читацьких уподобань. Фактично він є зваженим середнім TF-IDF-векторів тих історій, які вже зацікавили користувача. Якщо читач частіше лайкає або зберігає твори певної тематики, такі тексти сильніше впливають на його профіль. Після цього система порівнює профіль користувача з векторами інших опублікованих історій і визначає ступінь їхньої схожості за допомогою косинусної міри. Чим ближчим є текстовий профіль історії до профілю інтересів користувача, тим вищою буде її рекомендаційна оцінка [18].

Під час формування добірки алгоритм не повертає історії, які користувач уже бачив або з якими вже взаємодіяв. Із рекомендацій виключаються власні твори користувача, історії з лайками, закладками та записи, що вже присутні в його історії читання. Це важливо, оскільки рекомендаційна система має пропонувати новий контент, а не повторювати вже знайомі матеріали. Після фільтрації історії сортуються за оцінкою схожості, і користувачу повертаються ті, що мають найвищий підсумковий бал.

У системі також передбачено ситуацію, коли для персоналізації ще недостатньо інформації. Це стосується нових користувачів або тих, хто майже не взаємодіяв із контентом. Якщо сформувати профіль інтересів

неможливо, застосунок повертає популярні опубліковані історії. Вони відбираються за кількістю лайків і переглядів, а також із урахуванням дати створення. При цьому власні або вже переглянуті твори все одно виключаються, щоб список залишався корисним для конкретного користувача [19].

Для зменшення навантаження на сервер сформовані рекомендації зберігаються в таблиці recommendation_cache. Перед записом нової добірки попередні рекомендації користувача видаляються, після чого в таблицю додаються нові значення: ідентифікатор користувача, ідентифікатор історії, числова оцінка релевантності, версія алгоритму та час завершення актуальності кешу. У поточній версії застосовується алгоритм ml_tfidf_1.0, а строк життя збереженої добірки становить 6 годин. Це дозволяє не виконувати обчислення щоразу при відкритті головної сторінки, але водночас достатньо часто оновлювати рекомендації.

Обраний підхід є достатньо зрозумілим для реалізації та водночас дає змогу використати елементи машинного навчання в реальній логіці застосунку. Системі не потрібне окреме складне формування навчальної вибірки, адже вона спирається на характеристики вже опублікованих історій і дії користувачів, що поступово накопичуються. Для дипломного проєкту це практичне рішення: воно реалізоване в коді, інтегроване з базою даних і безпосередньо використовується у персональній стрічці рекомендацій.

Таким чином, рекомендаційна система у вебплатформі виконує дві основні функції. По-перше, вона спрощує пошук цікавого контенту для читача. По-друге, вона допомагає авторам отримувати додаткові перегляди від користувачів, яким їхні історії можуть справді підійти за змістом. Реалізація на основі TF-IDF та косинусної схожості є помірно складною з технічного погляду, але достатньо обґрунтованою й корисною для функціоналу системи.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональне тестування

Функціональне тестування вебзастосунку проводилося для перевірки того, чи коректно реалізовані основні можливості системи та чи відповідають вони вимогам, визначеним на попередніх етапах роботи. Під час перевірки увагу було зосереджено на найбільш важливих сценаріях використання: створенні облікового запису, вході до системи, роботі з історіями та главами, коментуванні, лайках, закладках, рекомендаціях, сповіщеннях, модерації та фіксації прогресу читання.

Тестування виконувалося вручну в браузері Google Chrome шляхом послідовного проходження основних користувацьких сценаріїв. Такий підхід дав змогу перевірити не лише окремі функції, а й узгодженість роботи інтерфейсу із серверною частиною. Окремо перевірялися повідомлення про помилки, коректність переходів між сторінками, оновлення даних без перезавантаження сторінки та реакція системи на некоректні дії користувача.

Першим було протестовано блок реєстрації та авторизації. Перевірялися створення нового акаунта з коректними даними, вхід із правильним логіном і паролем, спроба входу з помилковим паролем, реєстрація з уже зайнятою електронною поштою, а також надсилання порожніх форм. У результаті система коректно обробляла всі базові ситуації. При успішній реєстрації створювався новий користувач, при вході видавався JWT-токен, а в разі помилки відображалося зрозуміле повідомлення. Додатково було перевірено доступ до захищених сторінок без авторизації. Якщо токен був відсутній, користувач не міг виконати приватну дію та перенаправлявся до сторінки входу або отримував відповідну помилку від сервера.

Наступний етап стосувався створення та редагування історій. Було протестовано форму додавання нового твору із заповненням назви, опису, типу, жанрів і вікового обмеження. Окремо перевірялося завантаження обкладинки з допустимим форматом і спроби використання файлів, які не відповідають установленим обмеженням. Система правильно відхиляла некоректні файли та зберігала дозволені. Також було перевірено зміну інформації про вже створену історію, оновлення жанрів, статусу публікації та ознаки завершеності твору.

Після цього тестувалася робота з главами. Через текстовий редактор створювалася нова глава, додавалося форматування, виконувався запис у чернетки, а потім публікація. Було перевірено, що вміст зберігається без втрати структури та відображається на сторінці читання у відповідному вигляді. Окремо тестувалося редагування вже створеної глави, а також обмеження доступу до змін для сторонніх користувачів. Автор міг працювати лише зі своїм контентом, а спроба звернутися до чужого матеріалу через недозволений сценарій не давала можливості його змінити.

У межах перевірки взаємодії читачів із контентом тестувалися лайки, закладки та коментарі. Користувач міг поставити лайк історії, повторним натисканням його зняти, додати твір у закладки або прибрати з особистого списку. Для коментарів перевірялося створення нового повідомлення, відповідь на вже існуючий коментар, редагування власного тексту та його видалення. Система коректно оновлювала список коментарів і не дозволяла змінювати чужі записи. Також було перевірено лайки для коментарів та правильне оновлення відповідних лічильників. Приклад відображення сторінки читання глави та елементів взаємодії користувача з контентом наведено на рисунку 3.1.

Окрему увагу було приділено функції рекомендацій. Для користувача з накопиченою активністю перевірялося формування персональної добірки на головній сторінці. Після створення відповідних записів у кеші рекомендації відображалися коректно й відповідали даним, отриманим від серверної

частини. Також перевірялося, що в добірку не потрапляють уже переглянуті або власні історії користувача, якщо вони виключені умовами алгоритму. Для ситуацій із недостатньою історією взаємодій система повертала резервний список доступних творів.

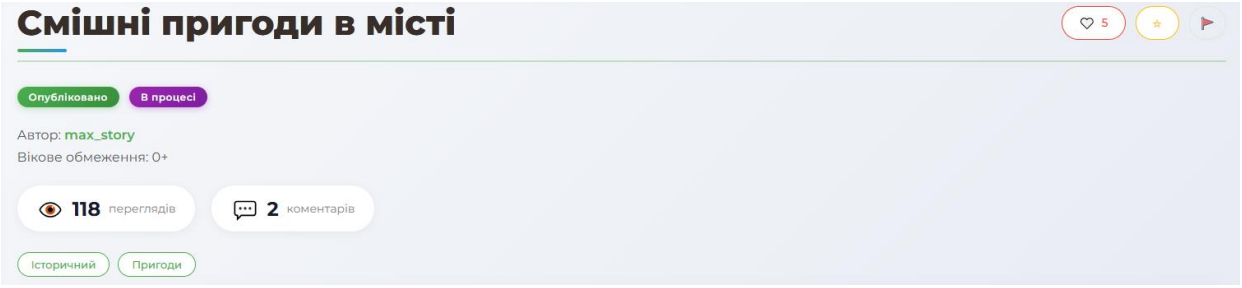


Рисунок 3.1 – Перевірка відображення сторінки читання глави та елементів взаємодії з контентом

Під час тестування сповіщень перевірялися події, що повинні створювати повідомлення для користувача. Зокрема, це надходження нового коментаря, відповідь на коментар, лайк історії, підписка та зміни, пов’язані з діями модератора. У верхній частині інтерфейсу правильно показувався лічильник непрочитаних сповіщень, а у випадному списку відображалися тексти повідомлень. Користувач міг позначити одне сповіщення як прочитане або оновити статус одразу для всіх.

У блоці модерації було перевірено подання скарги на історію та коментар, відображення звернень у панелі модератора, зміну статусу скарги, видалення контенту та блокування користувача. Тестування показало, що модератор має доступ до потрібних дій, тоді як звичайний користувач не може відкрити відповідний функціонал. Також було перевірено блокування акаунта на визначений строк і блокування без обмеження в часі. Після застосування обмежень користувач втрачав можливість виконувати авторизовані дії, що відповідало логіці системи.

Окремо перевірялася робота механізму відстеження читання. Під час перегляду глави система передавала на сервер дані про час перебування на сторінці та відсоток прокрутки. Після повернення до історії прогрес

відображався коректно. Також було перевірено ручне позначення глави як прочитаної та збереження цієї інформації в історії переглядів. У профілі користувача відображалися записи про прочитані або частково переглянуті історії, а групове видалення елементів зі списку працювало належним чином.

Крім цього, було протестовано пошук і фільтрацію. Перевірялися текстові запити за назвами історій і псевдонімами авторів, перехід до повної сторінки результатів, а також робота окремих параметрів каталогу: жанрів, вікових обмежень, типу твору, статусу завершеності й сортування. Активні фільтри можна було застосовувати, знімати окремо або скинути всі одразу, що робило поведінку інтерфейсу передбачуваною.

Також було проведено візуальну перевірку основних сторінок на різних розмірах екрана. Інтерфейс залишався придатним до використання як на широкому моніторі, так і на мобільних пристроях. Картки історій, форми, модальні вікна та таблиці коректно адаптувалися до вужчої ширини сторінки. У місцях, де вміст не міг зменшитися без втрати зручності, використовувався горизонтальний перегляд або компактніше компонування.

Під час перевірки також оцінювалася стійкість до типових помилкових дій. Система не приймала порожні обов’язкові поля, відхиляла файли непідтримуваного типу, не дозволяла виконувати захищені операції без авторизації та обмежувала доступ до адміністративних функцій. Використання параметризованих SQL-запитів і рольової перевірки з боку сервера сприяє безпечнішій роботі основних модулів.

За підсумками функціонального тестування можна зробити висновок, що ключові можливості вебплатформи працюють відповідно до поставлених вимог. Основні сценарії, передбачені для читача, автора та модератора, виконуються коректно. Під час перевірки було виявлено окремі незначні недоліки в поведінці інтерфейсу та відображенні даних на нестандартних розмірах екрана, після чого їх було виправлено.

3.2 Оцінка продуктивності та стабільності роботи системи

Окрім перевірки окремих функцій, під час тестування важливо було оцінити загальну стабільність роботи вебзастосунку. У межах цієї роботи основна увага приділялася не спеціалізованому навантажувальному тестуванню, а практичній перевірці того, як система поводиться під час звичайного використання: переходів між сторінками, виконання запитів до сервера, завантаження списків історій, роботи з профілем, коментарями, закладками та рекомендаціями.

Оцінювання проводилося в процесі ручного тестування застосунку на локальному середовищі розробки. Перевірялася робота основних сторінок і модулів під час послідовного виконання типових дій користувача. Зокрема, відкривалася головна сторінка, каталог історій, сторінка окремого твору, сторінка читання глави, профіль користувача, історія переглядів, модуль сповіщень і панель модератора. Окремо перевірялася поведінка системи після оновлення сторінки, повторного входу до акаунта та виконання кількох дій поспіль без перезапуску сервера.

У ході перевірки застосунок працював стабільно: сторінки відкривалися без критичних збоїв, дані після створення або редагування коректно оновлювалися, а переходи між розділами не призводили до зависання інтерфейсу. Запити на отримання історій, глав, коментарів, сповіщень і профільної інформації виконувалися коректно. Під час роботи з формами система належним чином реагувала на успішні операції та помилки, не залишаючи користувача без пояснення результату дії.

Окремо було перевірено стабільність взаємодії між клієнтською та серверною частинами. Після створення історії, додавання глави, постановки лайка або збереження твору в закладки зміни одразу відображалися в інтерфейсі. Після оновлення сторінки дані не втрачалися, що підтверджує коректне збереження інформації в базі даних. Аналогічно працювала система коментарів: нові повідомлення додавалися, відповіді відображалися у

відповідній гілці, а видалення власного коментаря не порушувало структуру списку.

Стабільність рекомендаційної системи перевірялася через запуск персональної добірки для користувача, який уже мав історію взаємодій із контентом. Після формування записів у таблиці кешу рекомендації відображалися на головній сторінці у відповідному блоці. Повторне відкриття сторінки не спричиняло помилок, а добірка залишалася доступною протягом строку дії кешу. Це показало, що рекомендаційний модуль правильно інтегрований із серверною частиною та фронтендом.

Також було звернено увагу на роботу системи після повторних дій користувача. Наприклад, повторне натискання на лайк змінювало його стан, а не створювало дубль. Аналогічно працювали закладки та підписки. Це свідчить про коректну обробку повторюваних операцій і збереження цілісності даних.

Під час перевірки модулів, які працюють із авторизацією, система також поводилася передбачувано. Якщо користувач не мав токена або його сесія вже не була дійсною, приватні запити не виконувалися. У таких ситуаціях сервер повертав помилку авторизації, а інтерфейс не надавав доступу до закритих функцій. Це важливо не лише з погляду безпеки, а й для стабільної поведінки системи в нетипових сценаріях.

Окремо оцінювалася робота вебзастосунку при відображенні більших списків даних. На сторінках історій, авторів, коментарів і сповіщень інтерфейс залишався придатним до використання, елементи не перекривали один одного, а прокручування сторінки працювало коректно. Для окремих списків застосовано пагінацію або поступове підвантаження, що зменшує навантаження на інтерфейс і робить перегляд інформації зручнішим.

Перевірка адаптивності показала, що основні сторінки коректно відображаються як на широкому екрані, так і на вузьких розмірах вікна браузера. Картки історій, форми, модальні вікна та навігаційні елементи змінюють розташування відповідно до доступного простору. Це дає змогу

користуватися застосунком не лише на настільному комп’ютері, а й на мобільних пристроях.

Повноцінне навантажувальне тестування із великою кількістю одночасних користувачів у межах цієї роботи не проводилося. Для такого дослідження доцільно застосовувати спеціальні інструменти, наприклад Locust або JMeter, а також виконувати перевірку вже на окремому серверному середовищі, наближеному до реальних умов експлуатації. Оскільки під час розробки система запускалася локально, отримані результати не можна коректно використовувати для точних висновків щодо максимальної кількості одночасних користувачів або граничного навантаження.

Водночас за результатами практичного тестування можна зробити висновок, що вебзастосунок працює стабільно в межах передбачених користувацьких сценаріїв, коректно обробляє основні запити, не втрачає дані після оновлення сторінок і правильно реагує на помилкові дії. Для дипломного проєкту цього достатньо, щоб підтвердити працездатність системи на рівні розробленого функціоналу. У разі подальшого розвитку платформи доцільно окремо провести навантажувальне тестування, перевірити поведінку сервера при великій кількості паралельних запитів і, за потреби, оптимізувати роботу з базою даних та кешуванням.

3.3 Оцінка ефективності рекомендаційної системи

Рекомендаційна система є важливою частиною вебзастосунку, оскільки вона впливає на зручність пошуку нового контенту. Якщо користувач бачить лише загальний каталог історій, йому доводиться самостійно переглядати велику кількість матеріалів. Персональні рекомендації частково розв’язують цю проблему, адже формують окрему добірку творів на основі попередньої активності читача.

У межах цієї роботи оцінювання рекомендаційної системи проводилося на рівні перевірки коректності її роботи та практичної доцільності отриманих результатів. Повноцінне статистичне дослідження із великою кількістю користувачів, окремою тестовою вибіркою та розрахунком спеціалізованих метрик не виконувалося, оскільки для цього потрібен значний масив реальних даних, накопичених у процесі тривалої експлуатації платформи. Тому основним завданням було переконатися, що алгоритм правильно формує персональні рекомендації, зберігає їх у базі даних і передає на фронтенд без помилок.

Перевірка здійснювалася на тестових облікових записах, для яких попередньо створювалася історія взаємодій із контентом. Користувач переглядав окремі історії, ставив лайки та додавав частину творів у закладки. Після цього запускався механізм формування рекомендацій. У результаті система створювала записи в таблиці recommendation_cache, де зберігалися ідентифікатор користувача, ідентифікатор рекомендованої історії, числовий показник відповідності, версія алгоритму та строк актуальності рекомендації.

Окремо перевірялося, чи відображаються сформовані добірки у відповідному блоці на головній сторінці. Після появи записів у кеші клієнтська частина успішно отримувала рекомендації через API та виводила їх у секції персональних історій. Це підтвердило, що серверний алгоритм, база даних і фронтенд працюють узгоджено. Приклад сформованої добірки рекомендованих історій наведено на рисунку 3.2.

Під час аналізу результатів зверталася увага на зміст рекомендованих історій. Алгоритм добирав твори, подібні до тих, з якими користувач уже взаємодіяв. Наприклад, якщо в історії активності переважали твори з певними жанрами та близьким описом, у рекомендаціях з’являлися історії зі схожим текстовим профілем. Такий результат відповідає принципу роботи реалізованого підходу, який базується на аналізі текстових характеристик історій і визначенні їхньої близькості до вподобань користувача.

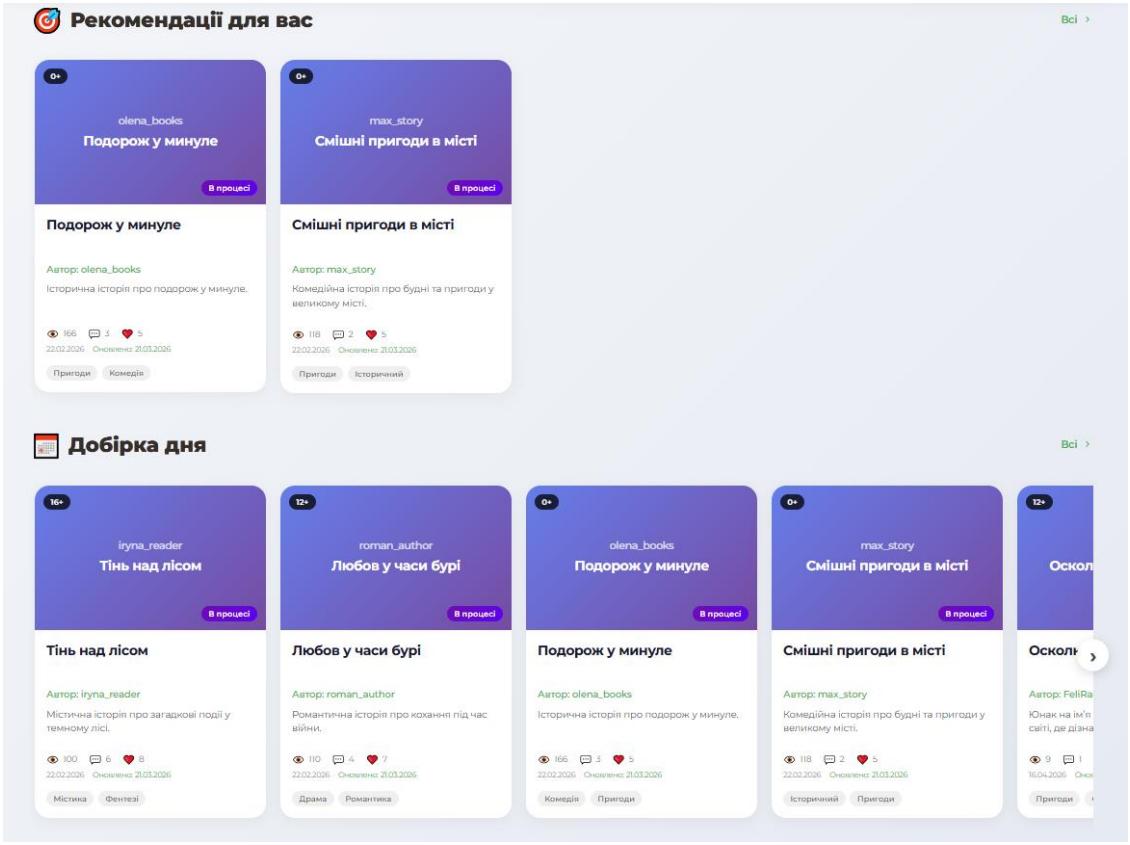


Рисунок 3.2 – Результат роботи рекомендаційного модуля вебплатформи

Також перевірявся сценарій із новим користувачем або користувачем, для якого ще немає достатньої кількості взаємодій. У цьому випадку система не завершує роботу з помилкою, а повертає резервний список історій, сформований без персоналізації. Це важливо для стабільності застосунку, оскільки персональні рекомендації мають коректно працювати не лише для активних читачів, а й для нових відвідувачів платформи.

Окрему увагу було приділено кешуванню результатів. Після формування добірки вона зберігалася на обмежений проміжок часу й могла бути використана повторно без негайного перерахунку. Такий підхід зменшує кількість обчислень і робить завантаження головної сторінки передбачуванішим. Під час перевірки повторне відкриття сторінки не спричиняло помилок, а рекомендації залишалися доступними протягом строку дії кешу.

За результатами тестування можна зробити висновок, що рекомендаційна система виконує своє основне призначення на

функціональному рівні. Вона формує персоналізовані списки історій для користувачів із наявною активністю, використовує резервний варіант за відсутності достатніх даних, коректно зберігає результати в базі даних і передає їх до клієнтської частини. Це підтверджує працездатність реалізованого механізму в межах розробленого вебзастосунку.

Для більш глибокої оцінки якості рекомендацій у майбутньому можна провести окреме дослідження після накопичення значнішої кількості реальних користувацьких дій. Тоді стане можливим обчислення показників точності рекомендацій, частоти переходів за ними та рівня подальшої взаємодії з рекомендованими історіями. У межах поточної роботи достатнім є підтвердження коректної реалізації алгоритму та його інтеграції з основним функціоналом платформи.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено вебплатформу для створення, публікації та читання авторських текстів. Реалізований застосунок орієнтований на дві основні групи користувачів: авторів, які створюють і розміщують власні твори, та читачів, які шукають цікаві історії, взаємодіють із контентом і стежать за оновленнями. Окремо передбачено функціонал модерації, що допомагає підтримувати порядок на платформі.

На початковому етапі було проаналізовано предметну область і розглянуто наявні рішення для публікації текстів, зокрема Wattpad, Webnovel, Royal Road, Medium та українську платформу «Аркуш». Порівняння цих сервісів дозволило визначити типові сильні й слабкі сторони подібних систем. Частина платформ має широкий функціонал і велику аудиторію, проте орієнтується переважно на міжнародний ринок. Інші краще враховують локальний контекст, але поступаються за зручністю інтерфейсу або кількістю можливостей для взаємодії. Це підтвердило доцільність розробки окремого вебзастосунку, який поєднує просту навігацію, базову соціальну взаємодію та інструменти для роботи з авторськими текстами.

На основі аналізу було сформульовано вимоги до системи та визначено основні ролі користувачів. У роботі враховано сценарії для гостя, читача, автора й модератора. Це дало змогу побудувати логіку застосунку послідовно, без суперечностей між функціями перегляду, створення контенту, взаємодії з творами та адміністративного контролю.

Під час проєктування було створено Use Case діаграму, сформовано клієнт-серверну архітектуру застосунку та спроєктовано базу даних. Структура даних охоплює користувачів, історії, глави, жанри, коментарі, лайки, закладки, підписки, сповіщення, скарги, історію читання, кеш рекомендацій та інші службові сутності. Така модель забезпечує збереження

всієї інформації, необхідної для роботи платформи, і підтримує взаємозв’язок між окремими функціональними модулями.

Серверну частину реалізовано на Python із використанням Flask, а клієнтську частину створено на React із Vite. Для зберігання даних використано MySQL. У системі реалізовано реєстрацію й авторизацію користувачів, керування профілем, створення історій і глав, завантаження обкладинок, публікацію та редагування текстів. Додатково впроваджено коментарі з відповідями, лайки, закладки, підписки, сторінки авторів, історію переглядів, сповіщення та панель модератора.

Окрему увагу було приділено безпеці. Паролі зберігаються у хешованому вигляді, доступ до закритих маршрутів контролюється через JWT-токени, а права користувачів розмежовуються відповідно до їхньої ролі. Для взаємодії з базою даних застосовано параметризовані SQL-запити, що робить обробку введених значень безпечнішою. У серверній логіці також передбачено перевірку блокування акаунтів і обмеження дій для користувачів, які порушили правила платформи.

Важливою частиною розробки стала рекомендаційна система. У поточній реалізації вона використовує підхід на основі TF-IDF-векторизації та косинусної схожості. Система формує добірку історій з урахуванням попередньої активності користувача та текстових характеристик творів. Результати тимчасово зберігаються в таблиці кешу, що дозволяє не виконувати однакові обчислення при кожному відкритті сторінки. Такий функціонал розширює можливості платформи й робить пошук нового контенту зручнішим.

Клієнтська частина забезпечує повний набір сторінок для основних сценаріїв роботи. Реалізовано головну сторінку з добірками історій, каталог із фільтрами, сторінку пошуку, профілі користувачів, сторінку окремої історії, режим читання глав, редактор текстів, налаштування акаунта, історію переглядів і панель модератора. Інтерфейс побудований на повторно використовуваних компонентах і адаптований до різних розмірів екрана.

У третьому розділі було виконано перевірку працездатності застосунку. Функціональне тестування охопило авторизацію, створення історій, редагування глав, коментування, лайки, закладки, сповіщення, модерацію, пошук і рекомендації. Окремо оцінювалася стабільність роботи системи під час типових дій користувача та коректність взаємодії між фронтендом, сервером і базою даних. За результатами перевірки основні функції працюють відповідно до поставлених вимог. Також було підтверджено коректне формування рекомендацій, їх запис у кеш і відображення на головній сторінці.

Практична цінність роботи полягає у створенні завершеного вебзастосунку, який може бути використаний як основа для реальної платформи авторських текстів. Проєкт уже містить не лише базові засоби публікації, а й інструменти для взаємодії аудиторії, відстеження активності та персоналізації контенту. Модульна структура дає змогу надалі розширювати функціонал без повного перегляду архітектури.

Подальший розвиток системи може передбачати вдосконалення пошуку, розширення можливостей рекомендаційної системи після накопичення більшої кількості користувацьких даних, реалізацію мобільної версії або окремого мобільного застосунку, додавання експорту історій у поширені формати, розширення статистики для авторів і поглиблення засобів модерації. За потреби в майбутньому також можна розглянути інтеграцію соціального входу та додаткові способи поширення контенту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kraxenberger M., Lauer G. Wreading on Online Literature Platforms. *Written Communication*. 2022. Vol. 39, No. 3. P. 462–496. URL : https://journals.sagepub.com/doi/10.1177/07410883221092730 (дата звернення : 16.05.2026).

2. Аркуш. Сучасна україномовна літературна платформа. URL : https://arkush.net/ (дата звернення : 16.05.2026).

3. Wattpad. Read & Write Stories. App Store. URL : https://apps.apple.com/ro/app/wattpad-read-write-stories/id306310789 (дата звернення : 16.05.2026).

4. WebNovel. Read Your Adventures. URL : https://www.webnovel.com/ (дата звернення : 16.05.2026).

5. Inkstone. WebNovel Author Platform. URL : https://inkstone.webnovel.com/ (дата звернення : 16.05.2026).

6. Royal Road. Home of Web Novels and Fan Fictions. URL : https://www.royalroad.com/ (дата звернення : 16.05.2026).

7. Medium. Writing and Publishing Your First Story. URL : https://help.medium.com/hc/en-us/articles/225168768-Writing-and-publishing-your-first-story (дата звернення : 16.05.2026).

8. Flask. Welcome to Flask Documentation. URL : https://flask.palletsprojects.com/ (дата звернення : 16.05.2026).

9. React. Describing the UI. URL : https://react.dev/learn/describing-the-ui (дата звернення : 16.05.2026).

10. Vite. Next Generation Frontend Tooling. URL : https://vite.dev/ (дата звернення : 16.05.2026).

11. MySQL. What is MySQL? URL : https://dev.mysql.com/doc/refman/9.0/en/what-is-mysql.html (дата звернення : 16.05.2026).

12. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. 2015. URL : https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення : 16.05.2026).

13. Flask-JWT-Extended. Documentation. URL : https://flask-jwt-extended.readthedocs.io/ (дата звернення : 16.05.2026).

14. Tiptap. Get Started with Tiptap Editor. URL : https://tiptap.dev/docs/editor/getting-started/overview (дата звернення : 16.05.2026).

15. Sass. Documentation. URL : https://sass-lang.com/documentation/ (дата звернення : 16.05.2026).

16. OWASP Foundation. Password Storage Cheat Sheet. URL : https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата звернення : 16.05.2026).

17. Scikit-learn. TfidfVectorizer. URL : https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (дата звернення : 16.05.2026).

18. Scikit-learn. Cosine Similarity. URL : https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html (дата звернення : 16.05.2026).

19. Pazzani M. J., Billsus D. Content-Based Recommendation Systems. In: The Adaptive Web. Springer, 2007. URL : https://link.springer.com/chapter/10.1007/978-3-540-72079-9_10 (дата звернення : 16.05.2026).

20. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. University of California, Irvine, 2000. URL : https://roy.gbiv.com/pubs/dissertation/top.htm (дата звернення : 16.05.2026).

ДОДАТОК А

Приклад коду рекомендацій

```
INTERACTION_WEIGHTS = {
    'like': 3.0,
    'bookmark': 2.5,
    'read_completed': 2.0,
    'read_partial': 1.0,
}
class RecommendationEngine:
    """
    Рекомендаційна система на основі машинного навчання.
    Використовується:
    - TF-IDF для векторизації текстових характеристик історій;
    - cosine similarity для порівняння профілю користувача з історіями;
    - лайки, закладки та історія читання для побудови профілю інтересів.
    """
    def __init__(self):
        self.conn = get_db_connection()
        self.cursor = self.conn.cursor(dictionary=True)
    def close(self):
        """Коректно закрити курсор і підключення до БД"""
        try:
            if self.cursor:
                self.cursor.close()
        except Exception:
            pass
        try:
            if self.conn:
                self.conn.close()
        except Exception:
```

```
        pass
# ОСНОВНИЙ МЕТОД
def get_ml_recommendations(self, user_id, limit=20):
    """
    Повернути персоналізовані ML-рекомендації для користувача.
    """
    try:
        limit = max(1, min(int(limit), 50))
        stories = self._get_published_stories()
        if not stories:
            return []
        interactions = self._get_user_interactions(user_id)
        # Якщо взаємодій ще немає, повертаємо популярні історії
        if not interactions:
            return self._get_popular_fallback(user_id, limit)
        story_ids = [story['id'] for story in stories]
        story_index = {
            story_id: index
            for index, story_id in enumerate(story_ids)
        }
        corpus = [
            self._build_story_text(story)
            for story in stories
        ]
        vectorizer = TfidfVectorizer(
            max_features=5000,
            ngram_range=(1, 2),
            lowercase=True
        )
        tfidf_matrix = vectorizer.fit_transform(corpus)
        user_profile = self._build_user_profile_vector(
            interactions=interactions,
            story_index=story_index,
            tfidf_matrix=tfidf_matrix
```

```
)
if user_profile is None:
    return self._get_popular_fallback(user_id, limit)
similarities = cosine_similarity(
    user_profile,
    tfidf_matrix
).flatten()
excluded_story_ids = self._get_excluded_story_ids(user_id)
ranked_items = []
for index, story in enumerate(stories):
    story_id = story['id']
    if story_id in excluded_story_ids:
        continue
    score = float(similarities[index])
    if score <= 0:
        continue
    ranked_items.append({
        'story_id': story_id,
        'score': score
   })
ranked_items.sort(
    key=lambda item: item['score'],
    reverse=True
)
ranked_items = ranked_items[:limit]
if not ranked_items:
    return self._get_popular_fallback(user_id, limit)
self._cache_recommendations(user_id, ranked_items)
ordered_story_ids = [
    item['story_id']
    for item in ranked_items
]
stories_data = self._get_stories_by_ids(ordered_story_ids)
score_map = {
```

```
            item['story_id']: item['score']
            for item in ranked_items
        }

        for story in stories_data:
            story['recommendation_score'] = round(
                score_map.get(story['id'], 0),
                4
            )
        stories_data.sort(
            key=lambda story: score_map.get(story['id'], 0),
            reverse=True
        )
      return stories_data
    except Exception as e:
        logger.error(f'Помилка ML-рекомендацій: {e}', exc_info=True)
        return self._get_popular_fallback(user_id, limit)
```