

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування факультету)

Комп'ютерні системи та мережі

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалаврський

(ступінь вищої освіти)

на тему: РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ВИДАВНИЧОЇ
ДІЯЛЬНОСТІ

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-512сп

Спеціальності

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(назва освітньої програми (спеціалізації))

ВОРОНЯНСЬКИЙ А.О.

(ПРИЗВИЩЕ та ініціали)

Керівник КАСЬЯН К.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЄФИМЕНКО М.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
 Кафедра комп'ютерних систем та мереж
 Ступінь вищої освіти бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри КУДЕРМЕТОВ Р.К.

«14» квітня 2025 року

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

ВОРОНЯНСЬКОГО Андрія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи для видавничої діяльності.

керівник проєкту (роботи) к. т. н., доцент, КАСЬЯН Костянтин Миколайович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “8” квітня 2025 року № 151

2. Строк подання студентом проєкту (роботи) 01.06.2025 р.

3. Вихідні дані до проєкту (роботи) існуючі методи та алгоритми розробки вебзастосунків; загальні принципи побудови комп'ютерних систем для публікаційної діяльності; технології HTML, CSS, JavaScript, PHP для реалізації інтерфейсу та серверної логіки; реляційні моделі даних і засоби проєктування баз даних; офіційна документація та довідкові матеріали щодо обраних мов програмування та фреймворків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз комп'ютерних систем для видавничої діяльності

Методи розробки комп'ютерної системи

Розробка комп'ютерної системи

Тестування та оптимізація комп'ютерної системи

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконання завдання
1-4	КАСЬЯН К.М., к.т.н., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання «14» квітня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Ознайомлення з темою роботи, постановка мети та завдань, узгодження структури	до 22.04.2025	
2	Аналіз існуючих систем для публікаційної діяльності, вивчення літературних джерел		
3	Формування вимог до функціоналу та архітектури комп'ютерної системи	до 24.04.2025	
4	Вибір мов програмування, бібліотек, фреймворків та інструментів розробки	до 28.04.2025	
5	Розробка бази даних та логічної структури системи	до 01.05.2025	
6	Реалізація клієнтської частини: створення інтерфейсу користувача	до 05.05.2025	
7	Реалізація серверної частини: обробка запитів, робота з базою даних	до 12.05.2025	
8	Інтеграція компонентів системи, тестування взаємодії модулів		
9	Проведення функціонального тестування та дослідження продуктивності	до 22.05.2025	
10	Оформлення пояснювальної записки, графічного матеріалу	до 25.05.2025	
11	Підготовка до захисту: редагування роботи, написання доповіді	до 01.06.2025	

Студент (ка)

_____ **Андрій ВОРОНЯНСЬКИЙ**
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ **Костянтин КАСЬЯН**
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
66 с., 15 рис., 12 табл., 15 джерел.

ВЕБСИСТЕМА, ВИДАВНИЧА ДІЯЛЬНІСТЬ, ПУБЛІКАЦІЯ, КОНТЕНТ,
СТАТТЯ, РЕДАГУВАННЯ, HTML, CSS, JAVASCRIPT, PHP, MYSQL,
ІНТЕРФЕЙС, АВТОР, АДМІНІСТРАТОР.

Об'єкт дослідження — процеси публікаційної діяльності у цифровому середовищі.

Метою дипломної роботи є створення сучасної комп'ютерної системи для видавничої діяльності, яка дозволяє автоматизувати процеси створення, редагування та управління текстовими матеріалами.

У процесі реалізації проєкту проведено огляд наявних рішень у сфері публікацій, виявлено їхні переваги й недоліки, визначено функціональні та нефункціональні вимоги до нової системи. Запропоновано архітектуру вебсистеми з поділом на клієнтську та серверну частини.

Система реалізована з використанням HTML, CSS, JavaScript для створення інтерфейсу користувача, PHP для серверної логіки, та MySQL для роботи з базами даних. Усі компоненти інтегровані у локальне середовище Open Server.

Дипломний проєкт містить опис основних модулів, структури бази даних, інтерфейсів користувача та адміністратора. Проведено функціональне тестування, яке підтвердило відповідність реалізованого функціоналу встановленим вимогам.

Запропоноване рішення є ефективним інструментом для організації видавничого процесу, що сприяє підвищенню якості, зручності та доступності обробки контенту. Перспективи подальшого розвитку включають розширення функціональності, підтримку нових форматів публікацій і вдосконалення взаємодії з користувачем.

ЗМІСТ

Вступ.....	6
1 Аналіз комп'ютерних систем та видавничої діяльності	8
1.1 Поняття комп'ютерного забезпечення для видавничої діяльності	8
1.2 Аналіз сучасних вебдодатків для видавничої діяльності.....	9
1.3 Вимоги до розробки комп'ютерної системи.....	13
1.4 Висновки до розділу 1	15
2 Методи розробки комп'ютерної системи	16
2.1 Структура вебдодатку	16
2.2 Розробка SRS	17
2.3 Аналіз технології розробки фронтенду.....	25
2.4 Аналіз технологій розробки бекенду	26
2.5 Висновки до розділу 2.....	27
3 Розробка комп'ютерної системи	28
3.1 Модулі комп'ютерної системи	28
3.2 База даних комп'ютерної системи.....	31
3.3 Локальний вебсервер	35
3.4 Опис інтерфейсу програмного забезпечення.....	37
3.5 Висновки до розділу 3.....	40
4 Тестування та Оптимізація комп'ютерної системи.....	41
4.1 Результати функціонального тестування	41
4.2 Експериментальне дослідження продуктивності та безпеки.....	43
4.3 Висновки до розділу 4.....	45
Висновки.....	47
Перелік джерел посилання	49
Додаток А Фрагмент коду	51

ВСТУП

У сучасному цифровому середовищі видавнича діяльність вимагає використання новітніх технологій для обробки, редагування та розповсюдження друкованих та електронних матеріалів. Зростання обсягів контенту та необхідність його оперативного управління стимулюють розробку комп'ютерних систем, які забезпечують автоматизацію ключових процесів видавничого циклу [1].

Застосування інформаційних технологій у видавничій сфері охоплює створення вебплатформ і систем керування публікаціями, що дозволяють організовувати спільну роботу над матеріалами, зберігати й структурувати контент, забезпечувати контроль версій, редагування, верстання та випуск кінцевого продукту [2].

До прикладів сучасних рішень належать системи керування контентом (CMS), редактори з підтримкою шаблонів та інтегровані середовища для підготовки друкованої продукції. Все це дає змогу покращити якість видавничого процесу, скоротити витрати часу й ресурсів, а також підвищити доступність публікацій [2].

Метою даної дипломної роботи є створення комп'ютерної системи для видавничої діяльності, яка забезпечує функції підготовки, редагування та керування видавничим контентом.

Предмет дослідження – комп'ютерна система, призначена для організації видавничих процесів.

Об'єкт дослідження – діяльність видавництва, пов'язана зі створенням та розповсюдженням друкованих та електронних матеріалів.

Основні задачі дипломної роботи:

- аналіз існуючих програмних рішень у видавничій сфері;
- формування вимог до нової комп'ютерної системи;
- проєктування архітектури програмного продукту;

- реалізація системи за допомогою сучасних технологій;
- перевірка функціональності та ефективності запропонованого рішення.

Реалізація комп'ютерної системи дозволить оптимізувати діяльність видавництв, забезпечити зручне середовище для обробки матеріалів, автоматизувати ключові операції та підвищити якість кінцевого продукту.

1 АНАЛІЗ КОМП'ЮТЕРНИХ СИСТЕМ ДЛЯ ВИДАВНИЧОЇ ДІЯЛЬНОСТІ

1.1 Поняття комп'ютерного забезпечення для видавничої діяльності

Поняття "комп'ютерне забезпечення для видавничої діяльності" включає в себе широкий спектр прикладних програм, розроблених для підтримки процесів створення, опрацювання, форматування та розповсюдження різноманітного інформаційного контенту, як-от книжки, журнали, газети, вебресурси тощо [1]. Основні категорії такого забезпечення поділяються на:

- системи для видавничих процесів – комплексні програмні рішення, що охоплюють повний видавничий цикл (наприклад, Adobe InDesign, QuarkXPress);
- інструменти для верстки сторінок – програми, що дозволяють компоувати текстові та графічні елементи у вигляді макетів (Microsoft Publisher, Scribus);
- текстові процесори – програми для підготовки й форматування текстових матеріалів (Microsoft Word, Google Docs, LibreOffice Writer);
- програми для графічного редагування – застосунки для створення та редагування ілюстрацій, зображень і логотипів (GIMP, Adobe Photoshop, Inkscape);
- CMS – системи управління контентом на сайтах і блогах (WordPress, Joomla, Drupal);
- редактори вебсторінок – програмне забезпечення для створення HTML/CSS-макетів і написання сценаріїв (Visual Studio Code, Adobe Dreamweaver);
- редактори PDF – програми для роботи з PDF-документами (Foxit, Adobe Acrobat).

Вибір конкретного інструменту залежить від функціональних потреб видавничої організації, її бюджету, а також рівня технічної компетенції персоналу [2].

У контексті наукових видавництв актуальність таких систем обумовлюється необхідністю ефективного розповсюдження результатів досліджень. Зростаючий обсяг наукового контенту потребує сучасних інтегрованих рішень, які не лише оптимізують процес публікації, а й забезпечують відповідність академічним вимогам, швидкий пошук, взаємодію між авторами, редакторами й рецензентами. Подібні системи часто включають в себе інструменти для редагування, коментування та перевірки матеріалів [3].

Однак більшість наявних рішень мають недоліки: складний інтерфейс, обмежена функціональність для спільної роботи, недостатня інтеграція з науковими базами та невисокий рівень безпеки обробки даних. Це знижує ефективність їх використання у професійному середовищі [1].

Сучасні вебсистеми, натомість, дають змогу швидко публікувати та рецензувати роботи, незалежно від географії користувача. Покращена взаємодія з іншими платформами та підтримка контролю змін, як версіонування документів, суттєво покращують якість редакційного процесу [4].

Інтеграція новітніх технологій, зокрема ШІ та машинного навчання, дозволяє створювати інструменти для автоматичного редагування, виявлення плагіату та покращення якості публікацій. Такі можливості важливі для динамічного розвитку видавничих систем.

Таким чином, актуальність розробки нової комп'ютерної системи полягає у врахуванні недоліків чинних рішень, створенні дружнього інтерфейсу, забезпеченні безпеки, розширенні інтеграційних можливостей і підтримці спільної роботи над публікаціями.

1.2 Аналіз сучасних вебдодатків для видавничої діяльності

Огляд сучасних вебдодатків, призначених для підтримки видавничого процесу [5].

Storyist — це потужний застосунок, орієнтований на авторів художніх та кіносценаріїв, який доступний на платформах MacOS та iOS. Програма забезпечує гнучке управління текстовими проєктами, допомагаючи користувачам будувати сюжетні лінії, створювати персонажів і формувати структуру творів. Серед її функцій:

- повноцінний текстовий редактор з підтримкою стилів, коментарів та режиму рецензування;
- інструменти для створення та редагування персонажів, локацій і сценарних сцен;
- шаблони, які відповідають стандартам оформлення романів і сценаріїв;
- синхронізація через хмарні сервіси та можливість роботи з різних пристроїв;
- функції експорту в популярні формати: PDF, ePub тощо.

Storyist ідеально підходить для творчих фахівців, які прагнуть зручності та організованості у процесі розробки проєкту [6].

На рисунку 1.1 представлено інтерфейс Storyist, де видно ключові елементи: панель сюжетних карток, вікно тексту, інструменти редагування, лічильник слів, мобільний додаток і навігаційні кнопки. Цей інтерфейс розроблено для забезпечення ефективної роботи над літературними й сценарними творами.



Рисунок 1.1 – Storyist [6]

Компанія Literature and Latte є розробником двох популярних інструментів для письменників: Scrivener і Scapple. Scrivener — багатофункціональна платформа для управління великими текстовими проєктами, яка дозволяє структурувати матеріали, зберігати нотатки, дослідження та фрагменти в одному середовищі [8] (рис. 1.2).

Scapple, у свою чергу, є простим візуальним редактором ідей, що дозволяє записувати думки та встановлювати між ними логічні зв'язки у форматі вільної мапи. Обидва інструменти сприяють креативному процесу та ідеально підходять для початкового планування і написання [5].

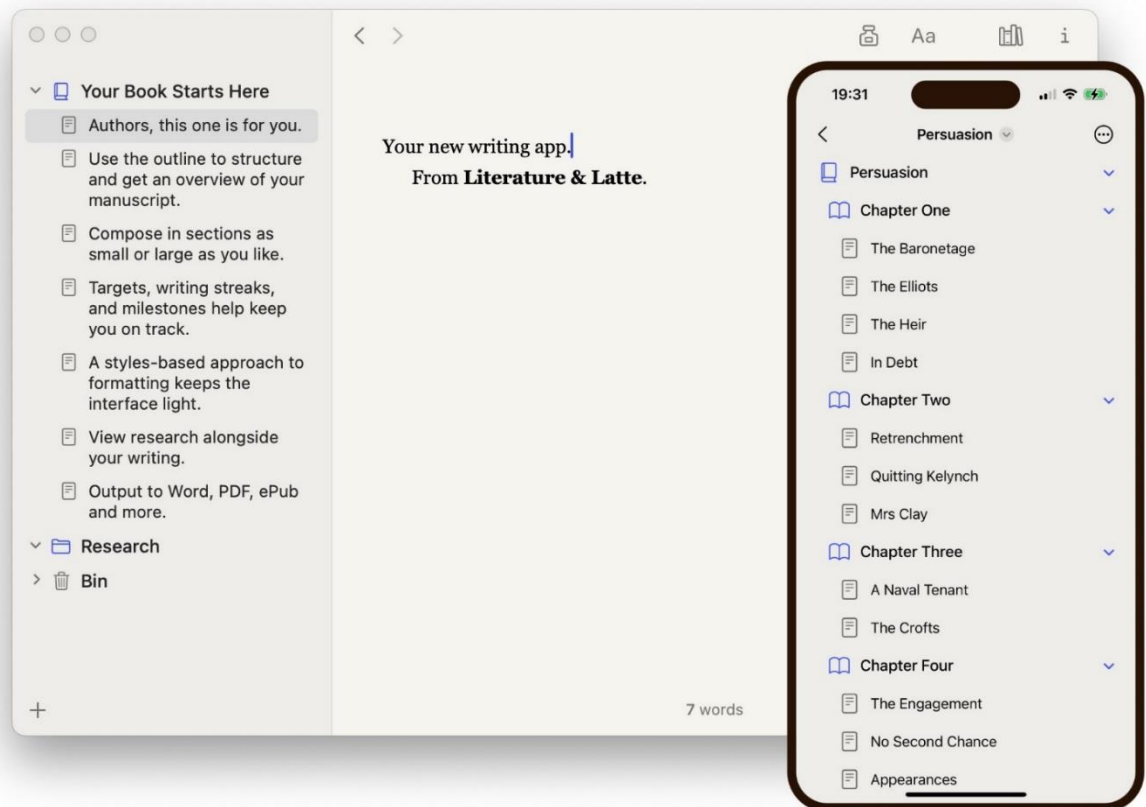


Рисунок 1.2 – Застосунок Scrivener [8]

Серед функціональних елементів, показаних на рисунку, є: огляд рукопису, компонування текстів за секціями, визначення цілей письма, підтримка стилів форматування, режим перегляду досліджень, інструменти експорту, зони зберігання дослідницьких матеріалів і кошик.

Ulysses — ще один інструмент, створений для авторів, які надають перевагу простоті інтерфейсу без зайвих відволікань. Програма підтримує синхронізацію через iCloud, використання розмітки Markdown, встановлення цілей письма, експортування у формати для друку чи онлайн-публікації (рис. 1.3).

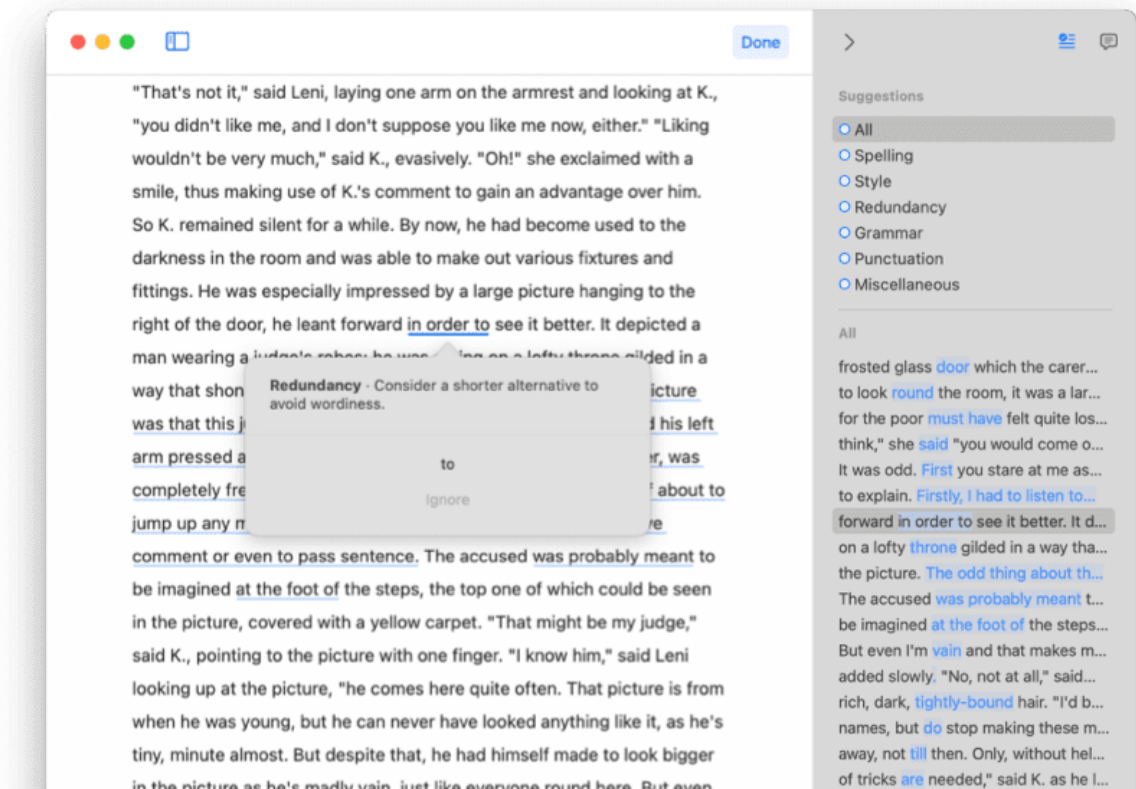


Рисунок 1.3 – Застосунок Ulysses.app [9]

Ulysses поєднує потужність, мінімалізм і функціональність, надаючи широкі можливості для написання та організації текстів.

Порівняльна таблиця 1.1 функціональності для Ulysses, Scrivener та Storyist.

Ця таблиця надає загальний огляд функціональності трьох популярних текстових редакторів, призначених для письменників і сценаристів. Варто зазначити, що всі три програми постійно оновлюються, і нові функції можуть бути додані з часом [10].

Таблиця 1.1 – Порівняння функціональності для Ulysses, Scrivener та Storyist:

Функція /Інструмент	Ulysses	Scrivener	Storyist
Текстовий редактор	Distraction-free, Markdown	Розширені опції форматування	Текстовий процесор з форматуванням стилю
Організація проекту	Групи і підгрупи, фільтри	Коркова дошка, обриси, папки і документи	Картки для персонажів, локацій, сюжетів
Синхронізація	iCloud	Dropbox (також підтримка інших служб)	iCloud
Експорт	PDF, DOCX, ePub, HTML, вебпублікації	PDF, DOCX, ePub, Kindle, RTF та інші	PDF, ePub, RTF, Final Draft
Шаблони документів	Шаблони для блогів та статей	Шаблони для романів, сценаріїв тощо	Шаблони для романів та сценаріїв
Ціль письма	Встановлення та відстеження цілей	Цілі сесій та проєктів	Відстеження прогресу через статистику
Інтерфейс	Мінімалістичний	Комплексний, налаштовуваний	Чистий та організований
Інструменти для рецензії	Коментарі та анотації	Розгорнуті коментарі, анотації, літературне рецензування	Коментарі та перегляд змін
Шаблони	Мінімалістичні та сучасні	Традиційні і гнучкі	Спеціалізовані для письменників і сценаристів
Автоматизація	AppleScript, URL-схеми	AppleScript, компіляція	Недоступна або обмежена
Мобільна версія	iOS	iOS	iOS
Візуалізація даних	Відсутня	Візуалізація даних через карти та обриси	Відсутня

1.3 Вимоги до розробки комп'ютерної системи

Процес створення комп'ютерного забезпечення для видавничої діяльності є складним та багаторівневим, що вимагає ретельного врахування потреб потенційних користувачів, зокрема авторів, редакторів та видавництв. Основне завдання – визначити функціонал, який забезпечить ефективну підтримку таких процесів, як редагування, керування чернетками, форматування текстів і підготовка до цифрової публікації [11].

Під час аналізу сучасних рішень важливо дослідити поточні тенденції у видавничій сфері та інтегрувати перевірені практики у власну розробку. Зокрема, варто застосовувати модульну архітектуру, яка дозволяє системі адаптуватися до змінних потреб користувачів. Окремі модулі повинні працювати незалежно та бути придатними до оновлення без порушення загальної функціональності.

Вибір технологій має вирішальне значення. На клієнтській стороні доцільно використовувати сучасні JavaScript-фреймворки, такі як React або Angular, що забезпечують гнучке створення динамічного інтерфейсу. Для серверної частини можуть бути застосовані PHP або Ruby, які забезпечують стабільність, захищеність та ефективність обробки запитів [8].

Важливим компонентом є впровадження високого рівня безпеки. Сюди входить надійна аутентифікація, авторизація, а також захист від таких загроз, як SQL-ін'єкції та XSS-атаки. Реалізація принципу «безпека з перших етапів» дозволяє зменшити кількість вразливостей на завершальних стадіях розробки [9].

Інтеграція із зовнішніми системами є необхідною умовою сучасних видавничих платформ. Зокрема, варто передбачити взаємодію з електронними комерційними сервісами, системами рецензування та хмарними сховищами. Для цього слід передбачити підтримку стандартів API і забезпечити зручні механізми обміну даними.

Крім функціональності, слід приділити увагу доступності. Інтерфейс має бути зрозумілим і адаптованим для людей з обмеженими можливостями. Це передбачає використання універсального дизайну, підтримку масштабування, клавіатурної навігації, озвучування тощо [10].

Завершальним етапом є тестування, що дозволяє перевірити надійність роботи програмного забезпечення в реальних умовах. Автоматизовані та регресійні тести мають стати частиною циклу розробки, що забезпечить оперативне виявлення помилок при оновленнях.

Окрім створення, система має передбачати механізми довгострокової підтримки – випуск оновлень, реагування на відгуки користувачів, усунення багів

та вдосконалення функціоналу. Це сприятиме високій якості продукту й задоволенню потреб кінцевих користувачів.

1.4 Висновки до розділу 1

У цьому розділі було розглянуто ключові поняття комп'ютерного забезпечення, яке використовується у видавничій діяльності, а також здійснено аналіз сучасних інструментів, орієнтованих на створення, редагування та структурування текстових матеріалів.

Ринок програмного забезпечення для видавничої сфери включає широкий набір інструментів – від базових текстових редакторів до спеціалізованих систем, які забезпечують повний цикл видавництва. Такі рішення, як Storyist, Scrivener і Ulysses, демонструють ефективні підходи до організації авторської роботи, дозволяючи структурувати документи, керувати персонажами, формувати текст і експортувати проекти у зручних форматах.

У результаті порівняння виявлено, що кожна з розглянутих платформ має власні переваги й недоліки, а вибір програмного засобу залежить від індивідуальних потреб користувача. Здобутий досвід дозволяє сформулювати основні характеристики, якими має володіти сучасна система для видавництва.

Зокрема, при створенні нового програмного продукту необхідно враховувати:

- наявність модульної структури, яка забезпечить гнучкість та масштабованість системи;
- вибір сучасних технологій для реалізації інтерфейсу та серверної логіки;
- високий рівень безпеки обробки даних;
- можливість інтеграції з іншими сервісами;
- дотримання принципів доступності для всіх категорій користувачів;
- налагоджений процес тестування та підтримки.

Таким чином, розробка комп'ютерної системи для видавничої діяльності має ґрунтуватися на вивченні актуальних потреб користувачів і тенденцій галузі. Вдало реалізоване рішення сприятиме підвищенню ефективності, зручності та якості процесів створення й розповсюдження контенту для редакторів, авторів та видавництв.

2 МЕТОДИ РОЗРОБКИ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Структура вебдодатку

Сьогодні вебзастосунки відіграють ключову роль у багатьох сферах суспільного життя. Вони базуються на моделі клієнт-сервер, де веббраузер виконує роль клієнта, а вебсервер — обробника запитів [11]. Схематичне зображення взаємодії компонентів вебдодатку показано на рисунку 2.1.



Рисунок 2.1 – Архітектура вебдодатку

У частині логіки, що виконується на стороні сервера, використовуються різноманітні технології. Однією з перших була CGI — Common Gateway Interface. Вона дала змогу генерувати динамічні вебсторінки та виступала посередником між браузером і сервером. CGI дозволяє адаптувати програми під різні серверні ОС.

Згідно з принципами CGI, кожен HTTP-запит запускає окремий процес для виконання відповідного скрипта. Такі скрипти можуть бути написані на Python, Perl, Tcl тощо. Спеціальний інтерпретатор приймає запит, виконує скрипт, а результат у вигляді HTML-коду надсилається клієнту [12].

Попри свою універсальність, CGI має обмеження в продуктивності, оскільки кожен запит створює новий процес. При великій кількості запитів це призводить до перевантаження системи.

Для вирішення цієї проблеми з'явилися Java Servlets — вони дозволяють обробляти запити в межах одного процесу, розподіляючи їх у потоках, що значно підвищує продуктивність. Сервлети мають платформну незалежність завдяки виконанню в JVM і широкому набору бібліотек.

Ще один підхід — JSP (Java Server Pages), який поєднує можливості Java з шаблонною розміткою, подібною до HTML чи PHP. JSP-компоненти компілюються в сервлети перед виконанням, що підвищує швидкість і стабільність [13].

Технологія .NET від Microsoft автоматизує керування ресурсами, обробку виключень та спрощує налагодження. Платформа підтримує використання мов із CLR, а вихідний код компілюється у проміжну мову, забезпечуючи кросплатформенність.

ASP.NET є прикладом реалізації сучасного середовища для веброзробки, підтримуючи незалежність від мови, обробку помилок, шаблонизовану генерацію сторінок і масштабування [14].

2.2 Розробка SRS

Вимоги до програмного забезпечення охоплюють сукупність функціональних і нефункціональних характеристик, які визначають поведінку, якість і обмеження системи. Для видавничої платформи ці вимоги формуються на

основі аналізу потреб редакторів, авторів, читачів, а також адміністраторів і технічних спеціалістів. Документ включає специфікації, діаграми варіантів використання, користувацькі сценарії та обмеження, що накладаються на реалізацію та інтерфейс.

Цільова аудиторія:

- читачі (відвідувачі) — неавторизовані користувачі, які переглядають контент, читають статті, залишають відгуки;
- автори (користувачі) — зареєстровані особи, які створюють і редагують публікації;
- адміністратори — особи, які керують вмістом, модерують коментарі, контролюють структуру розділів;
- технічна підтримка — користувачі з повним доступом до налаштувань безпеки, резервного копіювання та сервісів.

Функціональні вимоги.

Для класу користувачів «Читачі» (рис. 2.2.):

- перегляд списку публікацій;
- перегляд рейтингів публікацій;
- пошук за ключовими словами;
- коментування (анонімне або за email);
- реєстрація з email-підтвердженням.

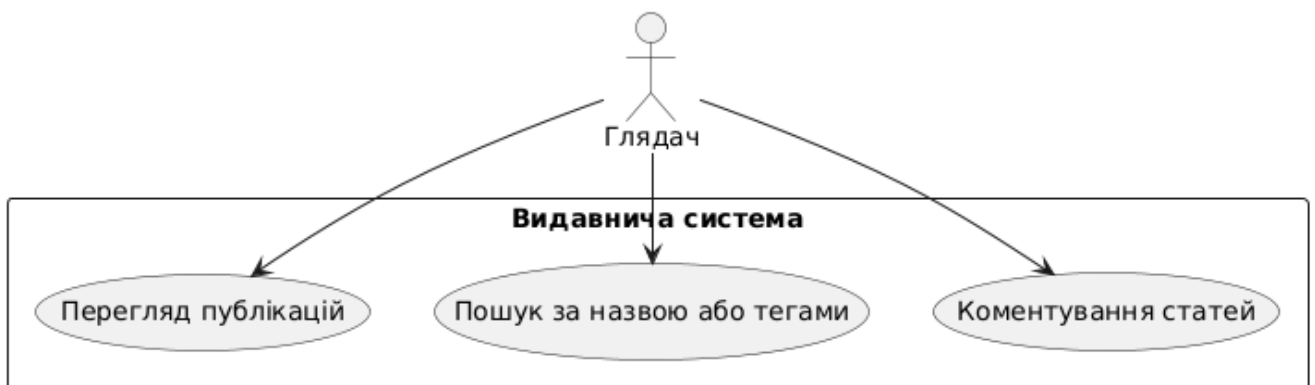


Рисунок 2.2– Діаграма варіантів використання «Глядач»

Для класу користувачів «Автори» (рис. 2.3):

- створення, редагування та публікація статей;
- додавання мультимедіа до публікацій;
- управління чернетками;
- перегляд історії публікацій;
- отримання статистики переглядів.



Рисунок 2.3– Діаграма варіантів використання «Автор»

Для класу користувачів «Адміністратори» (рис. 2.4):

- модерація коментарів;
- редагування та видалення публікацій;
- керування категоріями/тегами;
- рольовий доступ;
- перегляд журналів активності.



Рисунок 2.4 – Діаграма варіантів використання «Адміністратор»

Для класу користувачів «Технічна підтримка» (рис. 2.5, рис. 2.6):

- резервне копіювання та відновлення;
- керування базою даних;
- оновлення плагінів та зовнішніх бібліотек;
- управління візуальними шаблонами.



Рисунок 2.5 – Діаграма варіантів використання «Технічна підтримка»

Нефункціональні вимоги:

- сумісність з браузерами: Chrome, Firefox, Safari, Edge;
- адаптивність до мобільних пристроїв;
- швидкість завантаження: до 3 секунд;
- висока доступність (99,5% часу безвідмовної роботи);
- підтримка локалізації (українська, англійська);
- захист від XSS, CSRF та SQL-ін'єкцій;
- підтримка REST API.

Обмеження:

- реєстрація лише за допомогою email (без соціальних мереж);
- обов'язкова перевірка унікальності email;
- максимальний розмір зображення — до 2 МБ;
- паролі зберігаються лише в хешованому вигляді (bcrypt);
- публікації не можна редагувати після затвердження модератором.

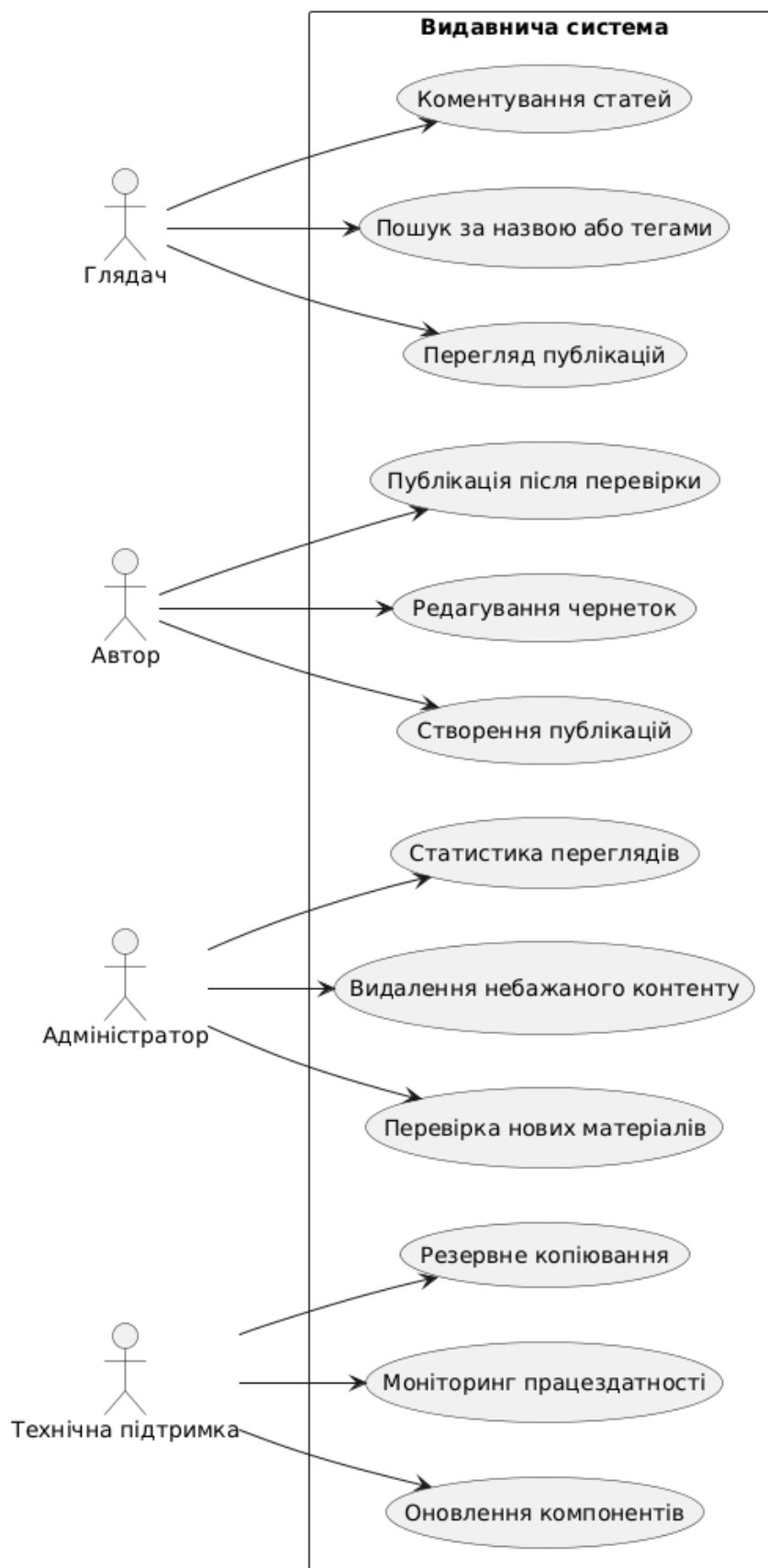


Рисунок 2.6 – Загальна діаграма варіантів використання видавничої системи

Користувач типу «Глядач» взаємодіє з видавничою платформою без попередньої авторизації. Після відкриття головної сторінки він ініціює запит на перегляд списку доступних публікацій. Інтерфейс платформи передає запит до серверної частини, яка звертається до бази даних і повертає результати користувачу. Глядач може скористатися пошуком, надіславши ключове слово, і отримати фільтровані результати. У разі бажання залишити коментар, користувач вводить його у форму, optionally вказуючи email. Запит зберігається в базі, а користувач отримує повідомлення про успішне надсилання або помилку (рис. 2.7).

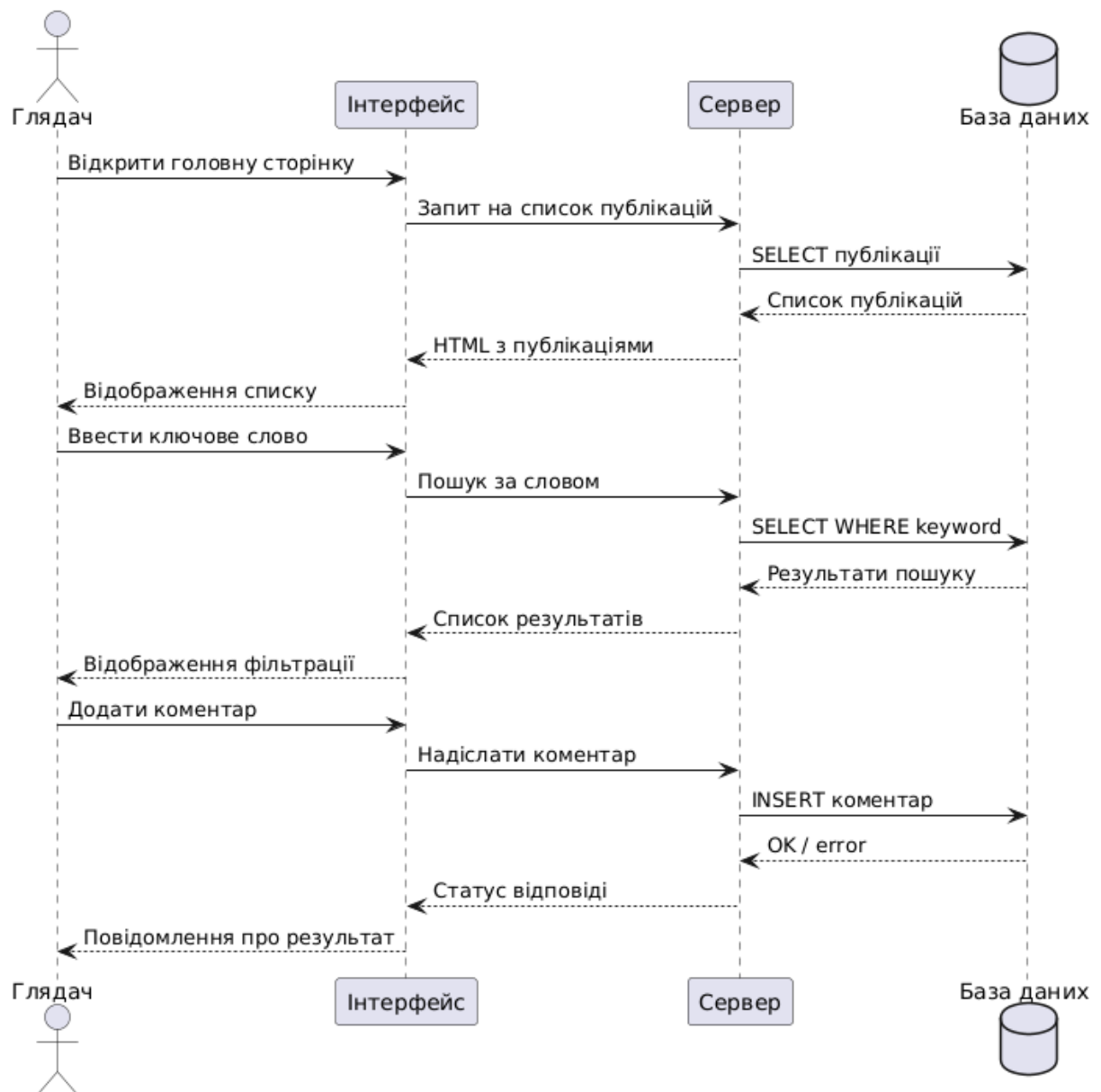


Рисунок 2.7 – Діаграма послідовності «Глядач»

Після успішної авторизації автор отримує доступ до особистого кабінету, де може створити нову публікацію. Користувач вводить заголовок, основний текст, додає зображення, після чого натискає кнопку «Зберегти як чернетку». Дані надсилаються на сервер і зберігаються в базі з відповідним статусом. Надалі автор може редагувати чернетку або надіслати її на модерацію. У відповідь система змінює статус публікації та повідомляє користувача (рис. 2.8).

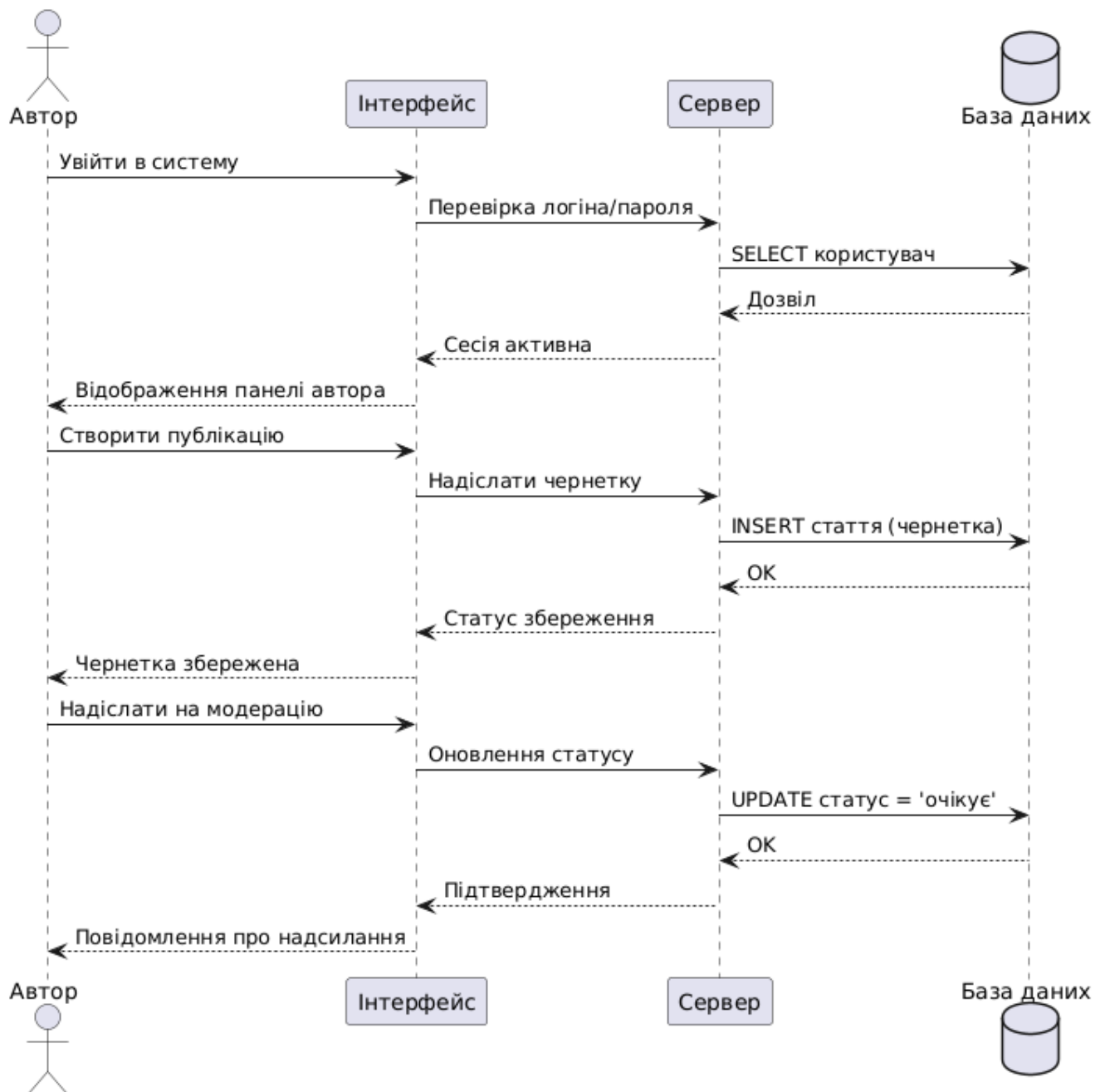


Рисунок 2.8 – Діаграма послідовності «Автор»

Після авторизації адміністратор відкриває панель модерації, де бачить список публікацій у статусі «очікує перевірки». Після перегляду вмісту він натискає кнопку «Схвалити» або «Відхилити». У першому випадку статус оновлюється до «опубліковано», у другому — публікація видаляється або редагується. Кожна дія фіксується у журналі (рис. 2.9).

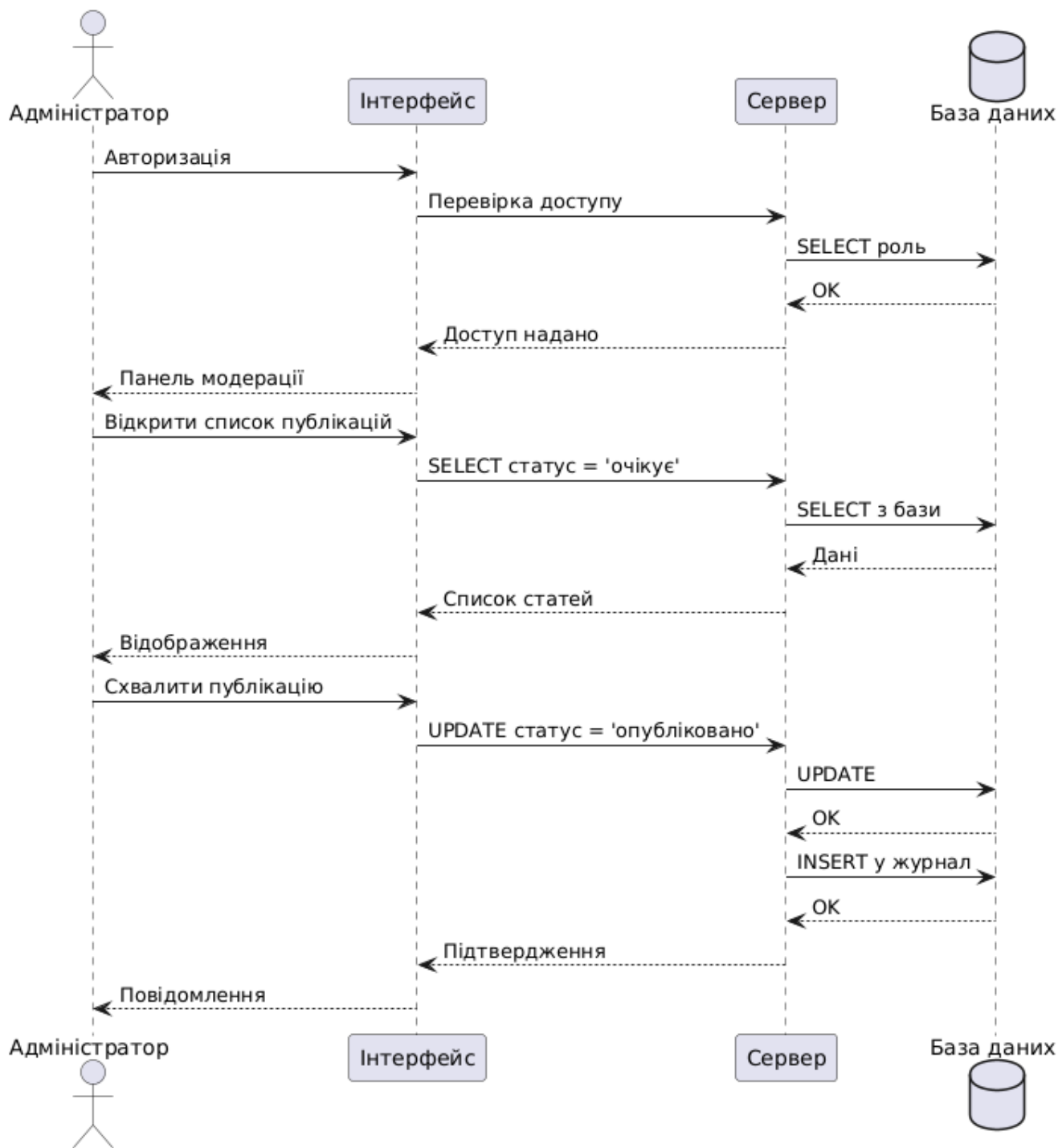


Рисунок 2.9 – Діаграма послідовності «Адміністратор»

2.3 Аналіз технології розробки фронтенду

Під час вибору технологій для побудови користувацького інтерфейсу слід враховувати низку параметрів: зручність у розробці, масштабованість, швидкодію та підтримку з боку спільноти. Односторінкові застосунки передбачають інтерактивність та високу реактивність, тому доцільно використовувати сучасні JavaScript-фреймворки. Найбільш популярними серед них є React, Angular і Vue.js (табл. 2.1).

React — бібліотека JavaScript, яку розробив Facebook. Вона базується на концепції компонентів, що забезпечує легке повторне використання коду. React відзначається високою продуктивністю та розвинутою екосистемою. Для управління станом використовуються додаткові бібліотеки, наприклад Redux.

Angular — це повноцінний фреймворк від Google для створення SPA-додатків. Він побудований на TypeScript, що дозволяє впроваджувати статичну типізацію й покращує контроль над логікою. Angular має потужні інструменти для маршрутизації, інжекції залежностей та валідації форм.

Vue.js — прогресивний фреймворк, який ставить акцент на простоті використання. Його легко інтегрувати у вже існуючі проекти, він підтримує реактивні дані й ефективно оновлює DOM. Vue ідеально підходить для невеликих та середніх застосунків.

Таблиця 2.1 - Порівняльна таблиця фреймворків

Фреймворк	Продуктивність	Масштабованість	Крива навчання	Екосистема
React	Висока	Висока	Середня	Велика
Angular	Висока	Висока	Висока	Велика
Vue.js	Висока	Середня	Низька	Зростаюча
Bootstrap	Висока	Висока	Середня	Велика

Зважаючи на особливості майбутньої комп'ютерної системи, було вирішено використовувати React. Ця бібліотека дозволяє розробляти масштабовані модульні інтерфейси з високою продуктивністю. Широка спільнота й багатий вибір ресурсів робить її зручною для впровадження й підтримки.

Окрім React, для реалізації інтерфейсу будуть застосовані: Redux для управління станом, React Router для маршрутизації, Axios для взаємодії з API та Styled Components або CSS-модулі для стилізації компонентів [10].

Комплексне використання цих інструментів забезпечить швидку, надійну та ефективну реалізацію сучасного вебінтерфейсу, що відповідатиме вимогам видавничої системи та очікуванням користувачів.

2.4 Аналіз технологій розробки бекенду

При виборі засобів реалізації серверної частини системи важливо враховувати критерії продуктивності, масштабованості, доступності бібліотек і досвіду команди розробки. Для побудови надійного серверного середовища розглядалися такі мови програмування як Python та JavaScript (через платформу Node.js), що представлені в таблиці 2.2.

Python є потужною, універсальною мовою з лаконічним синтаксисом, яка активно використовується для веброзробки. Найбільш популярними фреймворками є Django і Flask. Вони дозволяють швидко створювати повнофункціональні API, підтримують ORM, шаблонизування та інструменти для адміністрування [15].

Node.js дозволяє виконувати JavaScript на сервері, поєднуючи клієнтську та серверну частини застосунку в одній мові. Його неблокуюча модель введення-виведення забезпечує обробку великої кількості одночасних запитів. Express.js і Nest.js — популярні фреймворки, що полегшують побудову API та логіки додатку.

Таблиця 2.2 - Порівняльна таблиця технологій бекенду

Технологія	Продуктивність	Масштабованість	Екосистема	Крива навчання
Python	Висока	Висока	Велика	Низька
Node.js	Висока	Висока	Велика	Середня
PHP	Висока	Висока	Велика	Низька

У контексті розробки комп'ютерної системи для видавничої діяльності було обрано PHP, як мову з великою кількістю бібліотек для роботи з текстовим контентом, обробкою запитів і взаємодії з базами даних. PHP дозволяє швидко створити стабільний і функціональний бекенд.

Для забезпечення авторизації й автентифікації планується використання JSON Web Token (JWT), який дозволяє створювати захищені токени доступу. Це забезпечить контроль над сесіями користувачів та безпечну передачу інформації.

Для захисту системи від XSS, SQL-ін'єкцій та інших атак застосовуватимуться найкращі практики: фільтрація та екранування вхідних даних, параметризовані запити, контроль прав доступу.

Розгортання та масштабування системи відбуватиметься з використанням Docker-контейнерів та Kubernetes. Це надасть змогу організувати стабільне середовище, зручно переходити між етапами розробки, тестування й продакшену та забезпечить гнучке горизонтальне масштабування при зростанні навантаження.

2.5 Висновки до розділу 2

У цьому розділі було проведено всебічний аналіз методів та технологій розробки комп'ютерної системи для видавничої платформи. Розглянуто структуру вебдодатку, яка базується на архітектурі клієнт-сервер, та здійснено порівняльний аналіз серверних технологій від класичних CGI до сучасних фреймворків.

Було розроблено вимоги до програмного забезпечення (SRS), які охоплюють як функціональні, так і нефункціональні аспекти. Чітко визначено цільову аудиторію системи, що включає читачів, авторів, адміністраторів та технічну підтримку. Для кожної категорії користувачів сформовано детальні варіанти використання системи, які візуалізовано за допомогою UML-діаграм.

Аналіз технологій розробки фронтенду дозволив обґрунтувати вибір React як основної бібліотеки для побудови користувацького інтерфейсу. Вибір обумовлено високою продуктивністю цієї технології, можливістю компонентного підходу, широкою екосистемою та підтримкою з боку спільноти розробників.

При дослідженні технологій бекенду було проаналізовано переваги та недоліки різних мов та платформ, зокрема Python, Node.js та PHP. З урахуванням специфіки видавничої системи було обрано PHP як основну технологію серверної частини через її функціональність у роботі з текстовим контентом та широку екосистему бібліотек.

Окрему увагу приділено питанням безпеки, зокрема захисту від XSS, SQL-ін'єкцій та інших атак, а також організації авторизації за допомогою JWT. Для забезпечення масштабованості та спрощення розгортання системи заплановано використання контейнеризації на базі Docker та оркестрації з Kubernetes.

Таким чином, у розділі систематизовано та обґрунтовано технологічні рішення, які забезпечать розробку сучасної, надійної та масштабованої комп'ютерної системи для видавничої діяльності з урахуванням усіх функціональних та нефункціональних вимог.

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Модулі комп'ютерної системи

Комп'ютерна система, розроблена для підтримки видавничої діяльності, складається з низки функціональних модулів, кожен з яких відповідає за окремі

аспекти взаємодії користувача з системою. Ці модулі забезпечують зручне створення, редагування, збереження, перегляд та адміністрування публікаційного контенту. Архітектура системи побудована таким чином, щоб спростити масштабування та подальше обслуговування, забезпечуючи гнучкість, продуктивність і зручність користування.

Для реалізації функціоналу системи було використано набір сучасних інструментів та технологій, перелік яких узагальнено у таблиці 3.1. Серед них – програмні платформи, мови розмітки, мови програмування, інструменти керування базами даних, шаблони проєктування та локальне серверне середовище. Завдяки цьому система може працювати автономно, використовуючи сервер на локальній машині для тестування та демонстрації.

Таблиця 3.1 – Використані технології

Технологія	Опис
Google Chrome	Вебпереглядач
PhpStorm	Інтегроване середовище розробки
HTML	Мова розмітки
CSS	Мова стилів
Bootstrap	Фреймворк для розробки
PHP	Програмувальна мова
Active Record	Паттерн проєктування
ORM	Паттерн проєктування Object-Relational Mapping
Singleton	Паттерн проєктування
MySQL	Система управління базами даних
phpMyAdmin	Інструмент для роботи з базами даних
Open Server	Локальний сервер
Apache	Вебсервер

Зокрема, система побудована на основі мови програмування PHP, яка забезпечує обробку серверної логіки. У якості інструменту розробки

використовувався PhpStorm – інтегроване середовище, що дозволяє зручно працювати з HTML, CSS, JavaScript та PHP. Вебінтерфейс системи створено з використанням HTML і CSS, а також фреймворку Bootstrap, що дозволяє адаптувати зовнішній вигляд сторінок до різних розмірів екранів (табл. 3.2).

Таблиця 3.2 – Структура файлів

Директорія	Назва файлу	Опис
articles	create.php	Сторінка для створення нового матеріалу
	remove.php	Сторінка з інформацією про вилучення матеріалу
	update.php	Сторінка оновлення матеріалу
	display.php	Сторінка для перегляду матеріалу та коментарів
comments	modify.php	Сторінка для редагування коментарів
errors	auth_error.php	Сторінка помилки авторизації
	access_denied.php	Сторінка помилки доступу
	not_found.php	Сторінка помилки "Не знайдено"
	not_allowed.php	Сторінка помилки "Дія не дозволена"
	wrong_activation.php	Сторінка помилки невірному коду активації
	server_error.php	Сторінка помилки сервера
mail	activateAccount.php	Шаблон для підтвердження активності користувача
main	index.php	Головна сторінка з переліком усіх матеріалів

Для обміну даними з базою даних використовується шаблон Active Record, який дозволяє легко здійснювати операції з даними без необхідності написання складних SQL-запитів. У рамках взаємодії з БД застосовано підхід ORM (Object-Relational Mapping), що значно спрощує структурування запитів. Додатково у системі реалізовано шаблон Singleton, що забезпечує створення лише одного екземпляра певного класу, що важливо для ефективного керування конфігураційними параметрами.

Для локального розгортання серверної частини використано програмний комплекс Open Server. Цей набір інструментів дозволяє швидко запускати вебсервер Apache та взаємодіяти з базами даних MySQL через phpMyAdmin – графічний вебінтерфейс для управління таблицями, запитами та користувачами БД.

Щоб забезпечити зручну організацію коду, структура системи розділена на тематичні директорії. Такий підхід дозволяє розмежовувати логіку, подання та обробку даних за окремими функціональними блоками. Наприклад, окрема директорія містить файли, пов'язані з публікаціями (створення, оновлення, перегляд, видалення), інша – з коментарями, ще інша – зі сторінками обробки помилок, тощо.

Таким чином, кожен модуль виконує визначене завдання і є частиною загальної архітектури системи, що дає змогу ефективно реалізувати потреби користувачів – як адміністраторів, так і авторів публікацій. У наступних підрозділах буде детальніше розглянуто реалізацію бази даних, серверного середовища та інтерфейсу.

3.2 База даних комп'ютерної системи

База даних є основним елементом будь-якої інформаційної системи, що забезпечує зберігання, обробку та структурування інформації. У розробленій комп'ютерній системі для видавничої діяльності база даних виконує роль сховища ключових даних про публікації, користувачів, відгуки та процеси реєстрації. Грамотно побудована база даних є запорукою стабільності, швидкодії та масштабованості всієї системи.

Для реалізації бази даних використано реляційну систему керування базами даних MySQL. Ця СУБД була обрана завдяки своїй поширеності, високій

продуктивності, підтримці транзакцій, гнучкій роботі з індексами та великій кількості готових рішень.

Адміністрування структури бази даних, створення таблиць, керування записами та виконання SQL-запитів здійснюється за допомогою phpMyAdmin – інтуїтивно зрозумілого інструмента з графічним інтерфейсом. Цей вебдодаток дозволяє легко налаштовувати права доступу, переглядати вміст таблиць, імпортувати та експортувати дані, що є важливо для налагодження, тестування та супроводу системи.

Структура бази даних була спроектована з урахуванням логічних зв'язків між основними сутностями. Усі сутності представлені у вигляді окремих таблиць, між якими встановлено зв'язки відповідно до принципів нормалізації даних. Це дозволяє уникнути дублювання, забезпечити узгодженість інформації та полегшити її пошук.

Таблиця 3.3 – Опис структур бази даних

№	Назва	Опис	Особливості
1	publications	Таблиця публікацій	Доступ до додавання публікацій мають тільки адміністратори.
2	feedbacks	Таблиця відгуків	Всі зареєстровані користувачі можуть залишати відгуки.
3	account_details	Деталі облікових записів	При створенні облікового запису користувача його статус активації ініційований як 0, що вимагає підтвердження електронної пошти.
4	activation_keys	Ключі активації користувачів	Кожному новому користувачеві присвоюється хеш-ключ для верифікації пошти.

Таблиця 3.4 – Публікації

Атрибут	Тип даних	Опис
id	int	Ідентифікатор публікації
author_id	int	Ідентифікатор автора
title	varchar(255)	Заголовок публікації
content	text	Вміст публікації
created_at	Datetime	Дата створення

Таблиця 3.5 – Відгуки

Атрибут	Тип даних	Опис
id	int	Ідентифікатор відгука
user_id	int	Ідентифікатор користувача
publication_id	int	Ідентифікатор публікації
comment	text	Текст відгуку
date	datetime	Дата створення відгуку

Таблиця 3.6 – Деталі облікових записів

Атрибут	Тип даних	Опис
id	int	Ідентифікатор користувача
username	varchar(128)	Ім'я користувача
email	varchar(255)	Електронна адреса
is_active	tinyint(1)	Статус активації акаунта
role	enum('admin', 'user')	Роль користувача
password_hash	varchar(255)	Хеш паролю
auth_token	varchar(255)	Токен для авторизації
registered_at	datetime	Дата реєстрації

Таблиця 3.7 – Ключі активації

Атрибут	Тип даних	Опис
id	int	Ідентифікатор ключа
user_id	int	Ідентифікатор користувача
activation_code	varchar(255)	Активаційний код

Таблиця `publications` містить інформацію про створені публікації. Вона включає такі поля: унікальний ідентифікатор, заголовок, вміст публікації, ідентифікатор автора, дата створення. Ці дані дозволяють виводити список статей, здійснювати пошук за автором або назвою, сортувати матеріали за датою.

Таблиця `feedbacks` пов'язана з таблицями `publications` та `account\_details` через зовнішні ключі. Вона дозволяє зберігати коментарі користувачів до публікацій. Такий зв'язок реалізує модель “один до багатьох”, коли кожен користувач може залишити кілька коментарів, а кожна публікація може мати необмежену кількість відгуків.

У таблиці `account\_details` зберігається інформація про користувачів, включаючи ім'я, електронну адресу, хеш паролю, дату реєстрації, статус активності та роль (користувач чи адміністратор). Хешування паролю дозволяє зберігати його у зашифрованому вигляді, що підвищує безпеку системи. Використання поля `role` дозволяє організовувати контроль доступу до функціоналу.

Таблиця `activation\_keys` відповідає за процес активації облікових записів. Кожному новому користувачу при реєстрації генерується унікальний хеш-код, що надсилається на електронну пошту. Цей код записується в таблицю `activation\_keys` і зв'язується з користувачем через зовнішній ключ `user\_id`. Такий механізм дозволяє перевірити достовірність реєстрації (рис. 3.1).

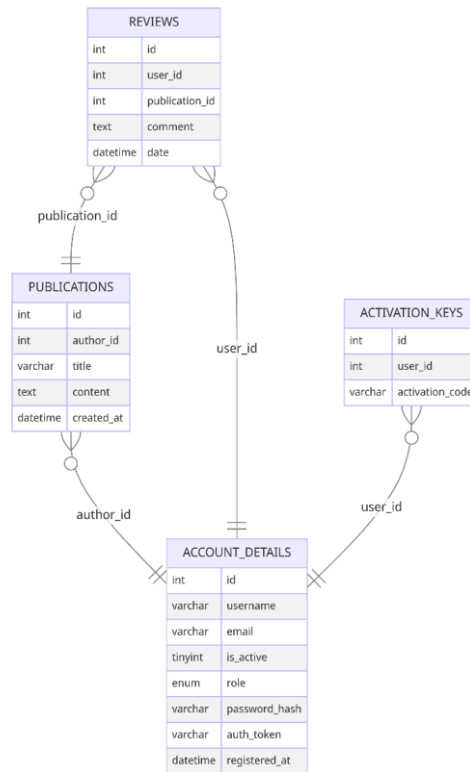


Рисунок 3.1 – Схема бази даних

На ER-діаграмі чітко відображено зв'язки між таблицями, типи зв'язків (1:1, 1:Б) та логіку побудови структури даних.

База даних є масштабованою – у разі необхідності до неї можна додати нові таблиці, наприклад, для збереження історії редагування публікацій, прикріплення файлів, ведення журналу активності тощо. Таким чином, розроблена структура відповідає як поточним потребам видавничої системи, так і дозволяє її подальше розширення без необхідності суттєвої перебудови.

3.3 Локальний вебсервер

Для забезпечення зручного середовища розробки та тестування комп'ютерної системи видавничого призначення було використано локальний вебсервер Open Server Panel. Це багатофункціональне середовище, створене

спеціально для потреб веброзробників, яке включає готові до використання серверні компоненти: вебсервер Apache, систему управління базами даних MySQL, а також інструменти адміністрування, зокрема phpMyAdmin.

Open Server має переваги, які роблять його ефективним інструментом у процесі локального розгортання програмних систем. Зокрема, він підтримує портативну установку, що дозволяє запускати сервер навіть із флеш-накопичувача, не потребуючи глибокої інтеграції у систему. Завдяки гнучким налаштуванням користувач може обирати версії PHP, конфігурувати компоненти серверного ПЗ, вмикати або вимикати модулі за потреби.

Середовище забезпечує централізоване керування сервісами, автоматичний запуск під час старту системи, журналювання подій, моніторинг стану серверів та виведення системних повідомлень. Усі ці інструменти значно полегшують процес тестування вебдодатків.

Встановлення Open Server відбувається шляхом завантаження архіву з офіційного сайту <https://ospanel.io>. Після розпакування та запуску програми в системному треї з'являється відповідна іконка, яка сигналізує про статус сервера. При першому запуску користувач має змогу вибрати компоненти, які буде активовано. Це стосується як серверних технологій (Apache, Nginx), так і супутніх служб (MySQL, PostgreSQL, Mail SMTP тощо).

Налаштування вебпроєкту здійснюється через конфігураційні файли та інтерфейс панелі керування. Open Server дозволяє створювати віртуальні хости, встановлювати шляхи до кореневих директорій проєктів, обирати порти, використовувати логування та перегляд журналів помилок.

У межах даного проєкту середовище Open Server було налаштоване таким чином (рис. 3.2):

- обрана версія PHP – 8.1, яка відповідає актуальним стандартам та має підтримку сучасних функцій;
- база даних MySQL 5.7 – стабільна версія для взаємодії з phpMyAdmin;
- активовано модуль Apache – як основний вебсервер для обробки HTTP-запитів;

- включено phpMyAdmin для зручного управління таблицями бази даних;
- створено віртуальний хост, прив'язаний до локального домену (наприклад, mypublisher.local).

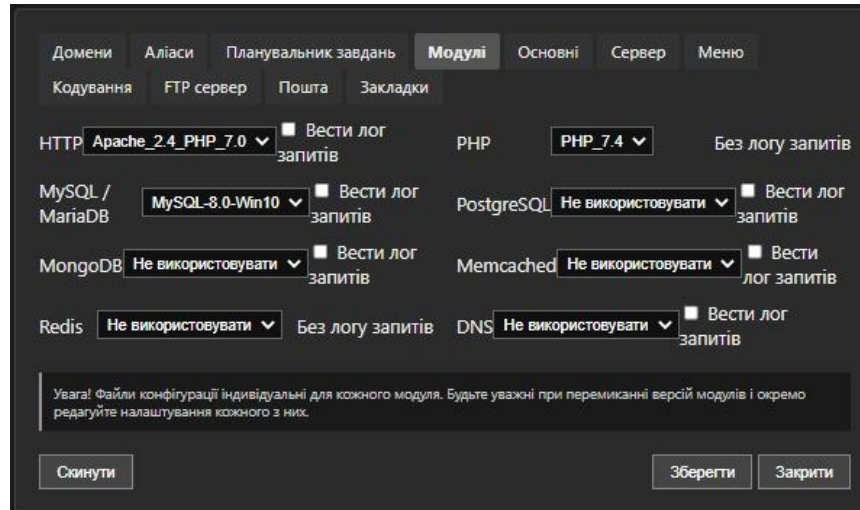


Рисунок 3.2 – Налаштування модулів локального вебсервера

Цей підхід дозволив відтворити умови, максимально наближені до реального хостингу, при цьому не залежачи від зовнішніх ресурсів. Завдяки такому налаштуванню розробник мав змогу одночасно створювати інтерфейс, писати серверну логіку, моделювати роботу бази даних і тестувати взаємодію між компонентами системи.

Локальний вебсервер є ключовим інструментом, який забезпечує автономну роботу програмного комплексу, знижує витрати на розробку, усуває проблеми, пов'язані з мережею, і сприяє швидшій ітерації змін у процесі створення функціоналу.

3.4 Опис інтерфейсу програмного забезпечення

Інтерфейс користувача є ключовим елементом взаємодії між системою та її користувачами. На рисунку 3.3 показано головну сторінку вебзастосунку, яка

служить центральною точкою доступу до публікацій. Тут користувач має змогу переглянути останні матеріали, а також швидко перейти до створення нових або редагування вже наявних статей.

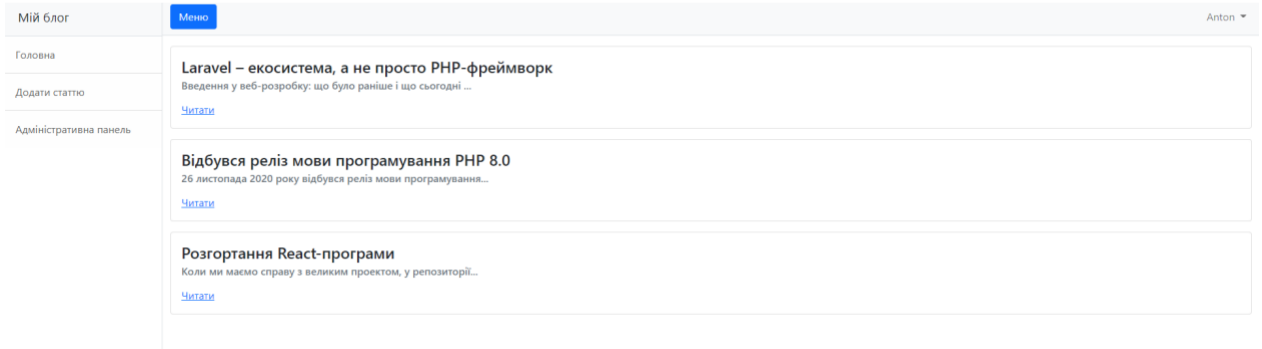


Рисунок 3.3 – Головна панель

Форма створення нового матеріалу надає зручні елементи для введення заголовка, основного тексту, завантаження мультимедійного контенту та вибору категорії. Це значно полегшує авторам підготовку публікацій (рис. 3.4, рис. 3.5).

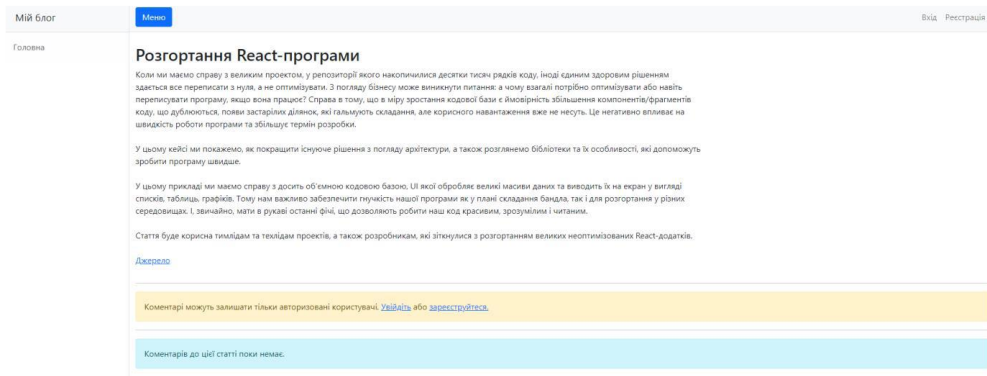


Рисунок 3.4 – Редагування публікації

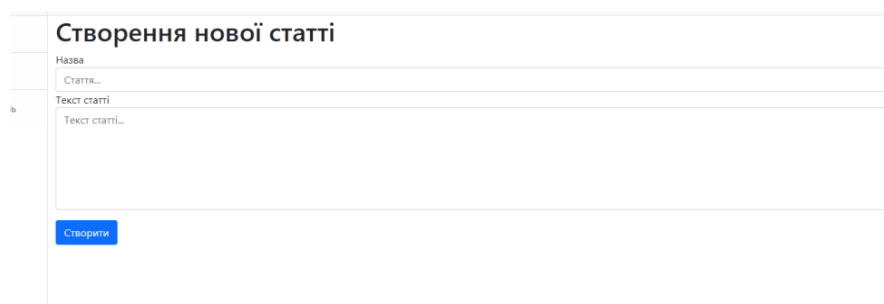


Рисунок 3.5 – Створення публікації

На рисунку 3.6 зображено інтерфейс адміністративної панелі. Цей розділ надає адміністраторам можливості моніторингу активності користувачів, перегляду аналітики, а також зміни параметрів системи для підтримання її стабільної роботи.



Рисунок 3.6 – Адміністративна панель

Рисунок 3.7 демонструє форму входу та реєстрації, які є необхідними для доступу до персоналізованих функцій. Інтерфейс авторизації містить поля для введення логіну та пароля, а також дозволяє створити новий обліковий запис із підтвердженням електронної пошти.

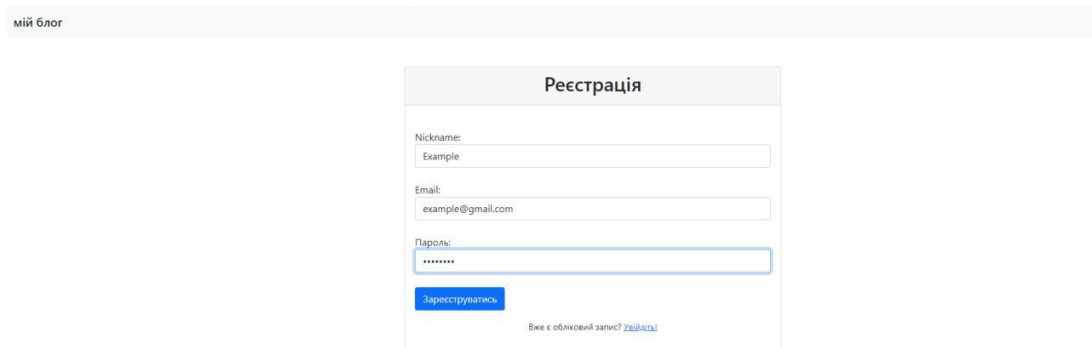


Рисунок 3.7 – Реєстрація користувачів

Таким чином, інтерфейс програмного забезпечення створено з урахуванням принципів доступності, зручності та функціональності, що забезпечує ефективну взаємодію користувача з системою як на рівні перегляду вмісту, так і його адміністрування.

3.5 Висновки до розділу 3

У третьому розділі було детально представлено процес розробки комп'ютерної системи для видавничої діяльності, що охоплює всі ключові аспекти її функціонування та технічної реалізації.

Розроблена система складається з взаємопов'язаних функціональних модулів, кожен з яких відповідає за окремі аспекти взаємодії користувача з платформою. Для реалізації системи було використано комплекс сучасних технологій, включаючи PHP як основну мову програмування, HTML і CSS для побудови інтерфейсу, Bootstrap для забезпечення адаптивності, а також шаблони проєктування Active Record, ORM та Singleton для ефективної організації коду та взаємодії з базою даних.

Спроектowana база даних MySQL складається з чотирьох взаємопов'язаних таблиць: `publications`, `feedbacks`, `account_details` та `activation_keys`, що забезпечують збереження та структурування даних про публікації, користувачів, відгуки та процеси реєстрації. Реляційна структура бази даних побудована з дотриманням принципів нормалізації, що забезпечує цілісність даних, уникнення дублювання інформації та підтримку можливості масштабування системи в майбутньому.

Для забезпечення середовища розробки та тестування було використано локальний вебсервер Open Server Panel, що включає необхідні компоненти: вебсервер Apache, СУБД MySQL та інструмент адміністрування phpMyAdmin. Це дозволило створити умови, максимально наближені до реального хостингу, при цьому забезпечуючи автономність розробки та зниження витрат на неї.

Розроблений інтерфейс користувача надає зручний доступ до всіх функцій системи, включаючи перегляд публікацій, створення та редагування матеріалів, адміністрування та авторизацію користувачів. Інтерфейс побудовано з урахуванням принципів ергономіки та адаптивності, що забезпечує комфортну роботу з системою на різних пристроях.

Таким чином, у результаті виконаної роботи було створено функціональну комп'ютерну систему, яка повністю відповідає сформульованим вимогам до програмного забезпечення для видавничої діяльності та забезпечує ефективну взаємодію між авторами, читачами та адміністраторами.

4 ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1 Результати функціонального тестування

Функціональне тестування є важливим етапом контролю якості програмного забезпечення, особливо для систем, що мають широкий набір функцій і орієнтовані на активну взаємодію користувачів. У контексті розробленої комп'ютерної системи для видавничої діяльності функціональне тестування дозволило перевірити, чи відповідає фактична поведінка основних компонентів очікуваному результату відповідно до заданих функціональних вимог.

Метою тестування було визначення працездатності ключових сценаріїв: створення та редагування публікацій, управління коментарями, реєстрація нових користувачів та перевірка адміністративного функціоналу. Для кожної з перевірених функцій було складено тест-кейси, які описують послідовність дій, очікуваний та фактичний результат.

Усі дії виконувалися в тестовому середовищі з попередньо підготовленою базою даних та активованим локальним вебсервером. Автори входили до системи під різними ролями (користувач, автор, адміністратор), що дозволило протестувати доступ до відповідних інтерфейсів (табл. 4.1).

Під час тестування було виявлено низку недопрацювань, які впливають на зручність та надійність системи:

- система не перевіряє обов'язковість заповнення полів при створенні публікації;

- користувачі не мають змоги редагувати або видаляти власні коментарі після їх додавання;
- відсутнє поле повторного введення паролю при реєстрації, що може призвести до помилок при авторизації.

Таблиця 4.1 – Функціональне тестування

№	Функція	Опис дій	Очікуваний результат	Фактичний результат
1	Створення нової публікації	Увійти в систему як автор; створити нову публікацію з заголовком, текстом та категорією; натиснути "Опублікувати".	Нова публікація створена та відображена на сайті.	Відповідає очікуваному.
2	Редагування публікації	Увійти в систему як автор; внести зміни в існуючу публікацію; натиснути "Зберегти зміни".	Зміни в публікації збережені та відображені.	Відповідає очікуваному.
3	Додавання коментаря	Увійти як користувач; написати коментар до публікації; натиснути "Відправити коментар".	Коментар доданий та відображений під публікацією.	Відповідає очікуваному.
4	Реєстрація нового користувача	Зареєструватися з ім'ям користувача, електронною поштою та паролем; активувати обліковий запис через електронну пошту.	Новий користувач зареєстрований та може увійти.	Відповідає очікуваному.

Для підвищення якості програмного продукту запропоновано:

- реалізувати валідацію форми публікації з виведенням повідомлень про помилки;
- створити інтерфейс редагування та видалення коментарів, доступний лише їх авторам;
- додати підтвердження паролю з перевіркою відповідності.

Впровадження запропонованих удосконалень дозволить не лише усунути наявні недоліки, а й підвищити загальну надійність системи та позитивне враження користувачів від взаємодії з програмним забезпеченням.

4.2 Експериментальне дослідження продуктивності та безпеки

Оцінка продуктивності та безпеки є важливим етапом впровадження будь-якої програмної системи, особливо якщо вона орієнтована на багатокористувацьку взаємодію. Розроблена комп'ютерна система для видавничої діяльності має забезпечувати не лише стабільну роботу при зростанні кількості користувачів, а й гарантувати захист персональних даних, автентифікацію доступу та збереження цілісності інформації.

У процесі експериментального тестування продуктивності було змодельовано ситуації підвищеного навантаження на систему. Було проведено серію вимірювань, під час яких до сервера одночасно надсилались запити від декількох користувачів, які здійснювали такі дії: авторизацію, створення публікацій, додавання коментарів, перегляд матеріалів. Паралельне виконання дій дало змогу оцінити, як система реагує на багатопоточні запити та навантаження на базу даних (табл. 4.2).

Результати вимірювання часу відгуку системи показали, що навіть при одночасному підключенні до 50 користувачів, час завантаження сторінок не перевищував 1,2 секунд, що відповідає сучасним вимогам до продуктивності

вебдодатків. Основні затримки були пов'язані із завантаженням великих зображень, що може бути оптимізовано шляхом додаткового стиснення та кешування.

Таблиця 4.2 - Результати дослідження продуктивності

Сценарій навантаження	Результати	Потенційні вузькі місця	Пропозиції щодо оптимізації
До 500 користувачів	Програмне забезпечення здатне обробляти без погіршення	-	-
1000 користувачів	Зростання часу відгуку та випадки відмови	Неоптимізовані запити до БД, надмірне використання ресурсів	Кешування, оптимізація запитів, асинхронні операції, налаштування сервера

Щодо безпеки, тестування охопило перевірку захищеності системи від поширених вебзагроз, зокрема:

- міжсайтове виконання скриптів (XSS);
- SQL-ін'єкції;
- brute force атаки на систему входу;
- перехоплення токенів сесії.

Було підтверджено, що у формі введення даних використовується фільтрація вводу, спеціальні символи екрануються, а запити до бази даних реалізовано за допомогою підготовлених виразів (prepared statements), що виключає можливість ін'єкцій.

Для захисту облікових записів реалізовано систему автентифікації з використанням токенів JWT. Кожен користувач після авторизації отримує

унікальний токен, який надсилається з кожним запитом у заголовку. Це дозволяє зберігати сесії без використання cookies та мінімізує ризики викрадення ідентифікаторів сесій.

Також перевірено ефективність системи управління правами доступу. Було виявлено, що користувачі не можуть отримати доступ до адміністративного інтерфейсу або редагувати чужі матеріали, що свідчить про коректно реалізовану рольову модель.

Таким чином, результати експериментального дослідження підтвердили, що система працює стабільно під навантаженням, демонструє хорошу швидкодію та має базовий рівень захисту від найбільш поширених атак. Це свідчить про надійність обраних архітектурних рішень і дозволяє рекомендувати систему для подальшого розгортання на реальних серверах.

4.3 Висновки до розділу 4

У ході проведеного функціонального тестування та експериментального дослідження продуктивності й безпеки комп'ютерної системи для видавничої діяльності було підтверджено її працездатність та виявлено напрямки для подальшого вдосконалення.

Функціональне тестування підтвердило успішне виконання ключових функцій системи: створення та редагування публікацій, управління коментарями, реєстрація користувачів та адміністрування. Виявлено декілька недоліків, зокрема відсутність валідації форм, неможливість редагування власних коментарів та недостатню перевірку при реєстрації користувачів. Запропоновано конкретні рішення для усунення цих недоліків.

Експериментальне дослідження продуктивності показало, що система здатна ефективно працювати при одночасному підключенні до 500 користувачів із прийнятним часом відгуку (до 1,2 секунд). Проте при навантаженні в 1000

користувачів спостерігалось зростання часу відгуку та випадки відмови, що вказує на необхідність оптимізації запитів до бази даних, впровадження кешування та асинхронних операцій.

Тестування безпеки підтвердило стійкість системи до поширених вебзагроз — XSS-атак, SQL-ін'єкцій та brute force атак. Реалізована JWT-автентифікація та рольова модель доступу забезпечують належний рівень захисту користувацьких даних та розмежування прав.

Таким чином, розроблена комп'ютерна система для видавничої діяльності відповідає поставленим функціональним вимогам, демонструє задовільну продуктивність при середньому навантаженні та має базовий рівень захисту від найбільш поширених типів атак. Виявлені недоліки та запропоновані шляхи оптимізації дозволять підвищити надійність та зручність системи перед її впровадженням у реальне середовище експлуатації.

ВИСНОВКИ

У межах бакалаврської роботи було реалізовано комп'ютерну систему для видавничої діяльності, що забезпечує ефективне управління створенням, редагуванням та публікацією текстових матеріалів. Система орієнтована на підтримку авторів і адміністраторів у процесі організації публікаційного контенту.

У першому розділі досліджено загальні принципи функціонування видавничих систем та проведено огляд сучасних вебрішень, що використовуються для роботи з текстами. Аналіз наявних систем дозволив окреслити функціональні потреби, визначити сильні сторони та виявити обмеження, які необхідно враховувати при створенні нового рішення.

Другий розділ присвячено формуванню вимог до функціоналу системи, вибору архітектурного підходу та підбору інструментів реалізації. Для клієнтської частини застосовано сучасні вебтехнології, а серверна логіка реалізована з використанням PHP. Також детально розглянуто питання безпеки, зберігання даних і підготовки середовища для запуску застосунку.

У третьому розділі подано опис основних компонентів системи, її бази даних, конфігурації локального сервера та користувацького інтерфейсу. Розглянуто функціональні особливості кожного модуля, що дає змогу повноцінно оцінити структуру розробленого рішення.

У четвертому розділі було проведено комплексну перевірку функціональності та стійкості системи. Тестування дозволило переконатися у відповідності функціоналу поставленим вимогам і виявити технічні аспекти, які потребують удосконалення. Експериментальне дослідження продуктивності підтвердило здатність системи працювати під навантаженням, а перевірка безпеки — її захищеність від базових атак. Запропоновано конкретні заходи для подальшої оптимізації.

Отже, створена комп'ютерна система для видавничої діяльності відповідає основним критеріям якості, є зручною у використанні та придатною для подальшого впровадження. Вона реалізована із застосуванням актуальних технологій, має модульну побудову, враховує питання безпеки та ефективності.

У перспективі система може бути розширена за рахунок нових функцій, інтеграції з зовнішніми сервісами, покращення інтерфейсу та автоматизації окремих процесів. Подальше тестування й оновлення дозволить підтримувати її на сучасному рівні та відповідати потребам користувачів у сфері видавничої діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Петренко А.В. Розробка вебдодатків з використанням React.js: навчальний посібник / А.В. Петренко. – Київ: Видавничий дім "Кондор", 2019. – 156 с.
2. Сергієнко А.О. Методи та інструменти розробки програмного забезпечення на Python: монографія / А.О. Сергієнко. – Львів: ЛНУ імені Івана Франка, 2020. – 192 с.
3. Ковальчук С.В. Сучасні технології веброзробки: від дизайну до реалізації / С.В. Ковальчук. – Харків: Форт, 2021. – 234 с.
4. Brown M.J. Introduction to JavaScript Full Stack Development: Create Complete Web Applications / M.J. Brown. – New York: Cambridge University Press, 2020. – 310 p.
5. Williams H.T. Practical Guide to Building an API Back End with Spring Boot / H.T. Williams. – San Francisco: Wiley, 2019. – 287 p.
6. Turner R. Advanced Techniques in Web Applications Development / R. Turner, J. White. – New York: Springer, 2022. – 253 p.
7. Федоров В.О. Основи використання Ruby on Rails у проєктах швидкої розробки / В.О. Федоров. – Київ: Академперіодика, 2019. – 174 с.
8. Morris K. Vue.js: Up and Running: Building Accessible and Performant Web Apps / K. Morris. – San Francisco: O'Reilly Media, 2018. – 210 p.
9. Thompson S. Web Development with Node and Express: Leveraging the JavaScript Stack / S. Thompson. – San Francisco: O'Reilly Media, 2021. – 335 p.
10. Гринь Д.М. Основи створення вебінтерфейсів з Angular / Д.М. Гринь. – Київ: УкрІнформ, 2020. – 188 с.
11. Walker T.E. Scalable Web Architecture and Distributed Systems / T.E. Walker. – New York: Apress, 2021. – 270 p.
12. Jenkins P. Building APIs with Django and Django Rest Framework / P. Jenkins. – San Francisco: Apress, 2019. – 300 p.

13. Кравченко Д.Б. Розробка багатомодульних систем на Java / Д.Б. Кравченко. – Харків: ХНУ імені В.Н. Каразіна, 2022. – 196 с.
14. Blackwell T. Modern Web Development: Understanding domains, technologies, and user experience / T. Blackwell. – New York: Pearson, 2018. – 250 p.
15. Заболотний І.П. Технології розробки програмного забезпечення / І.П. Заболотний, В.С. Кузьменко. – Львів: Літопис, 2021. – 202 с.

ДОДАТОК А

ФРАГМЕНТ КОДУ

Лістинг А.1 – Головна сторінка

```
<?php

/**
 * Клас для управління підключенням до бази даних
 * Реалізує паттерн Singleton
 */
class Database {
    private static $instance = null;
    private $connection;

    private $host = 'localhost';
    private $username = 'root';
    private $password = '';
    private $database = 'publishing_system';

    private function __construct() {
        try {
            $this->connection = new PDO(
                "mysql:host={$this->host};dbname={$this->database};charset=utf8",
                $this->username,
                $this->password,
                [
                    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
                ]
            );
        } catch (PDOException $e) {
            die("Помилка підключення до бази даних: " . $e->getMessage());
        }
    }

    public static function getInstance() {
        if (self::$instance === null) {
            self::$instance = new Database();
        }
        return self::$instance;
    }

    public function getConnection() {
        return $this->connection;
    }

    // Запобігання клонування
```

```

private function __clone() {}

// Запобігання серіалізації
public function __wakeup() {
    throw new Exception("Cannot unserialize singleton");
}
}

/**
 * Базовий клас Active Record для роботи з таблицями
 */
abstract class ActiveRecord {
    protected static $table;
    protected $attributes = [];
    protected $connection;

    public function __construct($attributes = []) {
        $this->connection = Database::getInstance()-
>getConnection();
        $this->attributes = $attributes;
    }

    public function __get($name) {
        return isset($this->attributes[$name]) ? $this-
>attributes[$name] : null;
    }

    public function __set($name, $value) {
        $this->attributes[$name] = $value;
    }

    public function save() {
        if (isset($this->attributes['id'])) {
            return $this->update();
        } else {
            return $this->insert();
        }
    }

    private function insert() {
        $fields = array_keys($this->attributes);
        $placeholders = ':' . implode(',', ':', $fields);
        $fieldsList = implode(', ', $fields);

        $sql = "INSERT INTO " . static::$table . " ({$fieldsList})
VALUES ({$placeholders})";
        $stmt = $this->connection->prepare($sql);

        foreach ($this->attributes as $key => $value) {
            $stmt->bindValue(":{key}", $value);
        }

        if ($stmt->execute()) {

```

```

        $this->attributes['id']          =          $this->connection-
>lastInsertId();
        return true;
    }
    return false;
}

private function update() {
    $fields = [];
    foreach ($this->attributes as $key => $value) {
        if ($key !== 'id') {
            $fields[] = "{$key} = :{$key}";
        }
    }

    $sql = "UPDATE " . static::$table . " SET " . implode(', ',
$fields) . " WHERE id = :id";
    $stmt = $this->connection->prepare($sql);

    foreach ($this->attributes as $key => $value) {
        $stmt->bindValue(":{key}", $value);
    }

    return $stmt->execute();
}

public static function find($id) {
    $db = Database::getInstance();
    $sql = "SELECT * FROM " . static::$table . " WHERE id =
:id";
    $stmt = $db->getConnection()->prepare($sql);
    $stmt->bindValue(':id', $id, PDO::PARAM_INT);
    $stmt->execute();

    $data = $stmt->fetch();
    return $data ? new static($data) : null;
}

public static function all($limit = null, $offset = 0) {
    $db = Database::getInstance();
    $sql = "SELECT * FROM " . static::$table;

    if ($limit) {
        $sql .= " LIMIT {$limit} OFFSET {$offset}";
    }

    $stmt = $db->getConnection()->prepare($sql);
    $stmt->execute();

    $results = [];
    while ($row = $stmt->fetch()) {
        $results[] = new static($row);
    }
}

```

```

        return $results;
    }

    public function delete() {
        if (!isset($this->attributes['id'])) {
            return false;
        }

        $sql = "DELETE FROM " . static::$table . " WHERE id = :id";
        $stmt = $this->connection->prepare($sql);
        $stmt->bindValue(':id', $this->attributes['id']);

        return $stmt->execute();
    }
}

<?php

require_once 'Database.php';

/**
 * Модель для роботи з публікаціями
 */
class Publication extends ActiveRecord {
    protected static $table = 'publications';

    public function __construct($attributes = []) {
        if (empty($attributes['created_at'])) {
            $attributes['created_at'] = date('Y-m-d H:i:s');
        }
        parent::__construct($attributes);
    }

    /**
     * Отримати всі публікації з інформацією про автора
     */
    public static function getAllWithAuthor($limit = 10, $offset =
0) {
        $db = Database::getInstance();
        $sql = "SELECT p.*, u.username as author_name
        FROM publications p
        JOIN account_details u ON p.author_id = u.id
        ORDER BY p.created_at DESC
        LIMIT :limit OFFSET :offset";

        $stmt = $db->getConnection()->prepare($sql);
        $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
        $stmt->bindValue(':offset', $offset, PDO::PARAM_INT);
        $stmt->execute();

        return $stmt->fetchAll();
    }
}

```

```

/**
 * Пошук публікацій за заголовком або змістом
 */
public static function search($query, $limit = 10) {
    $db = Database::getInstance();
    $sql = "SELECT p.*, u.username as author_name
            FROM publications p
            JOIN account_details u ON p.author_id = u.id
            WHERE p.title LIKE :query OR p.content LIKE :query
            ORDER BY p.created_at DESC
            LIMIT :limit";

    $stmt = $db->getConnection()->prepare($sql);
    $searchQuery = "%{$query}%";
    $stmt->bindValue(':query', $searchQuery);
    $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
    $stmt->execute();

    return $stmt->fetchAll();
}

/**
 * Отримати публікації конкретного автора
 */
public static function getByAuthor($authorId, $limit = 10) {
    $db = Database::getInstance();
    $sql = "SELECT * FROM publications
            WHERE author_id = :author_id
            ORDER BY created_at DESC
            LIMIT :limit";

    $stmt = $db->getConnection()->prepare($sql);
    $stmt->bindValue(':author_id', $authorId, PDO::PARAM_INT);
    $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
    $stmt->execute();

    $results = [];
    while ($row = $stmt->fetch()) {
        $results[] = new self($row);
    }

    return $results;
}

/**
 * Отримати кількість коментарів до публікації
 */
public function getCommentsCount() {
    $sql = "SELECT COUNT(*) as count FROM feedbacks WHERE
publication_id = :id";
    $stmt = $this->connection->prepare($sql);
    $stmt->bindValue(':id', $this->id);

```

```

$stmt->execute();

$result = $stmt->fetch();
return $result['count'];
}

/**
 * Отримати коментарі до публікації
 */
public function getComments() {
    $sql = "SELECT f.*, u.username
            FROM feedbacks f
            JOIN account_details u ON f.user_id = u.id
            WHERE f.publication_id = :id
            ORDER BY f.date ASC";

    $stmt = $this->connection->prepare($sql);
    $stmt->bindValue(':id', $this->id);
    $stmt->execute();

    return $stmt->fetchAll();
}

/**
 * Валідація даних публікації
 */
public function validate() {
    $errors = [];

    if (empty($this->title)) {
        $errors['title'] = 'Заголовок є обов\'язковим полем';
    } elseif (strlen($this->title) > 255) {
        $errors['title'] = 'Заголовок не може містити більше 255
символів';
    }

    if (empty($this->content)) {
        $errors['content'] = 'Зміст публікації є обов\'язковим
полем';
    } elseif (strlen($this->content) < 10) {
        $errors['content'] = 'Зміст публікації повинен містити
принаймні 10 символів';
    }

    if (empty($this->author_id)) {
        $errors['author_id'] = 'Автор публікації не вказаний';
    }

    return $errors;
}

/**
 * Скорочений опис публікації

```

```

    */
    public function getExcerpt($length = 150) {
        if (strlen($this->content) <= $length) {
            return $this->content;
        }

        return substr($this->content, 0, $length) . '...';
    }

    /**
     * Форматована дата створення
     */
    public function getFormattedDate() {
        return date('d.m.Y H:i', strtotime($this->created_at));
    }
}

<?php

require_once 'Database.php';

/**
 * Модель користувача
 */
class User extends ActiveRecord {
    protected static $table = 'account_details';

    const ROLE_USER = 'user';
    const ROLE_ADMIN = 'admin';

    public function __construct($attributes = []) {
        if (empty($attributes['registered_at'])) {
            $attributes['registered_at'] = date('Y-m-d H:i:s');
        }
        if (empty($attributes['role'])) {
            $attributes['role'] = self::ROLE_USER;
        }
        if (empty($attributes['is_active'])) {
            $attributes['is_active'] = 0;
        }
        parent::__construct($attributes);
    }

    /**
     * Створення нового користувача
     */
    public static function create($username, $email, $password) {
        $user = new self([
            'username' => $username,
            'email' => $email,
            'password_hash' => password_hash($password,
PASSWORD_DEFAULT),
            'is_active' => 0,

```

```

        'role' => self::ROLE_USER
    ]);

    $errors = $user->validate();
    if (!empty($errors)) {
        return ['success' => false, 'errors' => $errors];
    }

    if ($user->save()) {
        // Створюємо ключ активації
        $activationKey = bin2hex(random_bytes(32));
        ActivationKey::create($user->id, $activationKey);

        // Відправляємо email активації
        $user->sendActivationEmail($activationKey);

        return ['success' => true, 'user' => $user];
    }

    return ['success' => false, 'errors' => ['general' =>
'Помилка при створенні користувача']];
    }

/**
 * Валідація користувача
 */
public function validate() {
    $errors = [];

    // Перевірка імені користувача
    if (empty($this->username)) {
        $errors['username'] = 'Ім\'я користувача є
обов\'язковим';
    } elseif (strlen($this->username) < 3) {
        $errors['username'] = 'Ім\'я користувача повинно містити
принаймні 3 символи';
    } elseif ($this->isUsernameTaken()) {
        $errors['username'] = 'Це ім\'я користувача вже
зайняте';
    }

    // Перевірка email
    if (empty($this->email)) {
        $errors['email'] = 'Email є обов\'язковим';
    } elseif (!filter_var($this->email, FILTER_VALIDATE_EMAIL))
{
        $errors['email'] = 'Некоректний формат email';
    } elseif ($this->isEmailTaken()) {
        $errors['email'] = 'Користувач з таким email вже існує';
    }

    return $errors;
}

```

```

/**
 * Перевірка унікальності імені користувача
 */
private function isUsernameTaken() {
    $sql = "SELECT id FROM account_details WHERE username =
:username";
    if (isset($this->id)) {
        $sql .= " AND id != :id";
    }

    $stmt = $this->connection->prepare($sql);
    $stmt->bindValue(':username', $this->username);
    if (isset($this->id)) {
        $stmt->bindValue(':id', $this->id);
    }
    $stmt->execute();

    return $stmt->fetch() !== false;
}

/**
 * Перевірка унікальності email
 */
private function isEmailTaken() {
    $sql = "SELECT id FROM account_details WHERE email =
:email";
    if (isset($this->id)) {
        $sql .= " AND id != :id";
    }

    $stmt = $this->connection->prepare($sql);
    $stmt->bindValue(':email', $this->email);
    if (isset($this->id)) {
        $stmt->bindValue(':id', $this->id);
    }
    $stmt->execute();

    return $stmt->fetch() !== false;
}

/**
 * Автентифікація користувача
 */
public static function authenticate($username, $password) {
    $db = Database::getInstance();
    $sql = "SELECT * FROM account_details WHERE username =
:username AND is_active = 1";
    $stmt = $db->getConnection()->prepare($sql);
    $stmt->bindValue(':username', $username);
    $stmt->execute();

    $userData = $stmt->fetch();

```

```

        if ($userData && password_verify($password,
$userData['password_hash'])) {
            return new self($userData);
        }

        return null;
    }

    /**
     * Генерація JWT токена
     */
    public function generateToken() {
        $header = json_encode(['typ' => 'JWT', 'alg' => 'HS256']);
        $payload = json_encode([
            'user_id' => $this->id,
            'username' => $this->username,
            'role' => $this->role,
            'exp' => time() + (24 * 60 * 60) // 24 години
        ]);

        $headerEncoded = str_replace(['+', '/', '='], ['-', '_',
        ''], base64_encode($header));
        $payloadEncoded = str_replace(['+', '/', '='], ['-', '_',
        ''], base64_encode($payload));

        $signature = hash_hmac('sha256', $headerEncoded . "." .
        $payloadEncoded, 'your-secret-key', true);
        $signatureEncoded = str_replace(['+', '/', '='], ['-', '_',
        ''], base64_encode($signature));

        return $headerEncoded . "." . $payloadEncoded . "." .
        $signatureEncoded;
    }

    /**
     * Активація акаунта
     */
    public function activate($activationCode) {
        $key = ActivationKey::findByUserAndCode($this->id,
        $activationCode);
        if (!$key) {
            return false;
        }

        $this->is_active = 1;
        $result = $this->save();

        if ($result) {
            $key->delete();
        }

        return $result;
    }
}

```

```

/**
 * Відправка email активації
 */
private function sendActivationEmail($activationKey) {
    $to = $this->email;
    $subject = 'Активация акаунта';
    $message = "
    <html>
    <head>
        <title>Активация акаунта</title>
    </head>
    <body>
        <h2>Вітаємо, {$this->username}!</h2>
        <p>Дякуємо за реєстрацію у нашій видавничій системі.</p>
        <p>Для активації акаунта перейдіть за посиланням:</p>
        <a
href='http://localhost/activate.php?code={$activationKey}'>Активуват
и акаунт</a>
        <p>Якщо ви не реєструвалися на нашому сайті,
проігноруйте це повідомлення.</p>
    </body>
    </html>
    ";

    $headers = "MIME-Version: 1.0" . "\r\n";
    $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";
    $headers .= 'From: noreply@publishingsystem.local' . "\r\n";

    mail($to, $subject, $message, $headers);
}

/**
 * Перевірка ролі адміністратора
 */
public function isAdmin() {
    return $this->role === self::ROLE_ADMIN;
}

/**
 * Перевірка активності акаунта
 */
public function isActive() {
    return $this->is_active == 1;
}
}

/**
 * Модель ключів активації
 */
class ActivationKey extends ActiveRecord {
    protected static $table = 'activation_keys';
}

```

```

        public static function create($userId, $code) {
            $key = new self([
                'user_id' => $userId,
                'activation_code' => $code
            ]);
            return $key->save();
        }

        public static function findByUserAndCode($userId, $code) {
            $db = Database::getInstance();
            $sql = "SELECT * FROM activation_keys WHERE user_id =
:user_id AND activation_code = :code";
            $stmt = $db->getConnection()->prepare($sql);
            $stmt->bindValue(':user_id', $userId);
            $stmt->bindValue(':code', $code);
            $stmt->execute();

            $data = $stmt->fetch();
            return $data ? new self($data) : null;
        }
    }
}

<?php
session_start();
require_once '../models/User.php';
require_once '../models/Publication.php';
require_once '../includes/auth_check.php';

// Перевірка авторизації
if (!isset($_SESSION['user_id'])) {
    header('Location: ../errors/auth_error.php');
    exit;
}

$user = User::find($_SESSION['user_id']);
if (!$user || !$user->isActive()) {
    header('Location: ../errors/access_denied.php');
    exit;
}

$errors = [];
$success = false;

// Обробка форми
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $title = trim($_POST['title'] ?? '');
    $content = trim($_POST['content'] ?? '');

    // Створення нової публікації
    $publication = new Publication([
        'title' => $title,
        'content' => $content,
        'author_id' => $user->id
    ]);

```

```

    });

    // Валідація
    $errors = $publication->validate();

    if (empty($errors)) {
        if ($publication->save()) {
            $success = true;
            $_SESSION['message'] = 'Публікацію успішно створено!';
            header('Location: display.php?id=' . $publication->id);
            exit;
        } else {
            $errors['general'] = 'Помилка при збереженні
публікації';
        }
    }
}
?>

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Створення нової публікації</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstra
p.min.css" rel="stylesheet">
    <link href="../../assets/css/custom.css" rel="stylesheet">
</head>
<body>
    <!-- Навігація -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand"
href="../../main/index.php">Видавнича система</a>
            <div class="navbar-nav ms-auto">
                <a class="nav-link"
href="../../main/index.php">Головна</a>
                <a class="nav-link" href="create.php">Створити
статтю</a>
                <?php if ($user->isAdmin()): ?>
                    <a class="nav-link"
href="../../admin/panel.php">Адмін панель</a>
                <?php endif; ?>
                <a class="nav-link" href="../../auth/logout.php">Вихід
(<?= htmlspecialchars($user->username) ?>)</a>
            </div>
        </div>
    </nav>

    <div class="container mt-4">

```

```

<div class="row justify-content-center">
  <div class="col-md-8">
    <div class="card">
      <div class="card-header">
        <h4 class="mb-0">Створення нової
публікації</h4>
      </div>
      <div class="card-body">
        <?php if (!empty($errors['general'])): ?>
          <div class="alert alert-danger">
            <?=
htmlspecialchars($errors['general']) ?>
          </div>
        <?php endif; ?>

        <?php if ($success): ?>
          <div class="alert alert-success">
            Публікацію успішно створено!
          </div>
        <?php endif; ?>

        <form method="POST" action="">
          <div class="mb-3">
            <label for="title" class="form-
label">Заголовок публікації</label>
            <input
              type="text"
              class="form-control"
              id="title"
              name="title"
              value="<?=
htmlspecialchars($_POST['title']) ?>"
              placeholder="Введіть заголовок
публікації"
              required
            >
            <?php if (isset($errors['title'])):
?>
              <div class="invalid-feedback">
                <?=
htmlspecialchars($errors['title']) ?>
              </div>
            <?php endif; ?>
          </div>

          <div class="mb-3">
            <label for="content" class="form-
label">Зміст публікації</label>
            <textarea
              class="form-control"
              id="content"
              <?=
isset($errors['content']) ? 'is-invalid' : '' ?>

```

```

                                name="content"
                                rows="15"
                                placeholder="Введіть текст
публікації..."
                                required
                                ><?=
htmlspecialchars($_POST['content'] ?? '') ?></textarea>
                                <?php                                if
(isset($errors['content'])): ?>
                                <div class="invalid-feedback">
                                    <?=
htmlspecialchars($errors['content']) ?>
                                </div>
                                <?php endif; ?>
                                <div class="form-text">
                                    Мінімальна довжина: 10 символів
                                </div>
                                </div>
                                <div class="d-grid gap-2 d-md-flex
justify-content-md-end">
                                <a href="../main/index.php"
class="btn btn-secondary me-md-2">Скасувати</a>
                                <button type="submit" class="btn
btn-primary">Опублікувати</button>
                                </div>
                                </form>
                                </div>
                                </div>
                                </div>

<!-- Поради для авторів -->
<div class="card mt-4">
    <div class="card-header">
        <h5 class="mb-0">Поради для написання</h5>
    </div>
    <div class="card-body">
        <ul class="list-unstyled">
            <li><i class="text-primary">✓</i>
Використовуйте зрозумілий та лаконічний заголовок</li>
            <li><i class="text-primary">✓</i>
Структуруйте текст на абзаци</li>
            <li><i class="text-primary">✓</i>
Перевіряйте орфографію та пунктуацію</li>
            <li><i class="text-primary">✓</i>
Додавайте конкретні приклади та факти</li>
        </ul>
    </div>
</div>
</div>
</div>
</div>

```

```

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.
bundle.min.js"></script>
<script>
    // Автозбереження чернетки в localStorage
    const titleInput = document.getElementById('title');
    const contentTextarea = document.getElementById('content');

    function saveDraft() {
        const draft = {
            title: titleInput.value,
            content: contentTextarea.value,
            timestamp: new Date().getTime()
        };
        localStorage.setItem('publication_draft',
JSON.stringify(draft));
    }

    function loadDraft() {
        const savedDraft =
localStorage.getItem('publication_draft');
        if (savedDraft) {
            const draft = JSON.parse(savedDraft);
            // Завантажуємо чернетку тільки якщо форма порожня
            if (!titleInput.value && !contentTextarea.value) {
                if (confirm('Знайдено збережену чернетку.
Завантажити її?')) {
                    titleInput.value = draft.title || '';
                    contentTextarea.value = draft.content || '';
                }
            }
        }
    }

    // Збереження чернетки кожні 30 секунд
    setInterval(saveDraft, 30000);

    // Збереження при введенні тексту
    titleInput.addEventListener('input', saveDraft);
    contentTextarea.addEventListener('input', saveDraft);

    // Завантаження чернетки при завантаженні сторінки
    document.addEventListener('DOMContentLoaded', loadDraft);

    // Очищення чернетки при успішній публікації
    <?php if ($success): ?>
        localStorage.removeItem('publication_draft');
    <?php endif; ?>
</script>
</body>
</html>

```

Національний університет «Запорізька політехніка»
Кафедра комп'ютерних систем та мереж

Розробка комп'ютерної системи для видавничої діяльності.

Виконав студент групи КНТ-512сп
Керівник, к.т.н, доцент

Воронянський А.О.
КАСЬЯН К.М.

Мета та задачі

Метою даної дипломної роботи є створення комп'ютерної системи для видавничої діяльності, яка забезпечує функції підготовки, редагування та керування видавничим контентом.

Предмет дослідження - комп'ютерна система, призначена для організації видавничих процесів.

Об'єкт дослідження - діяльність видавництв, пов'язана зі створенням та розповсюдженням друкованих та електронних матеріалів.

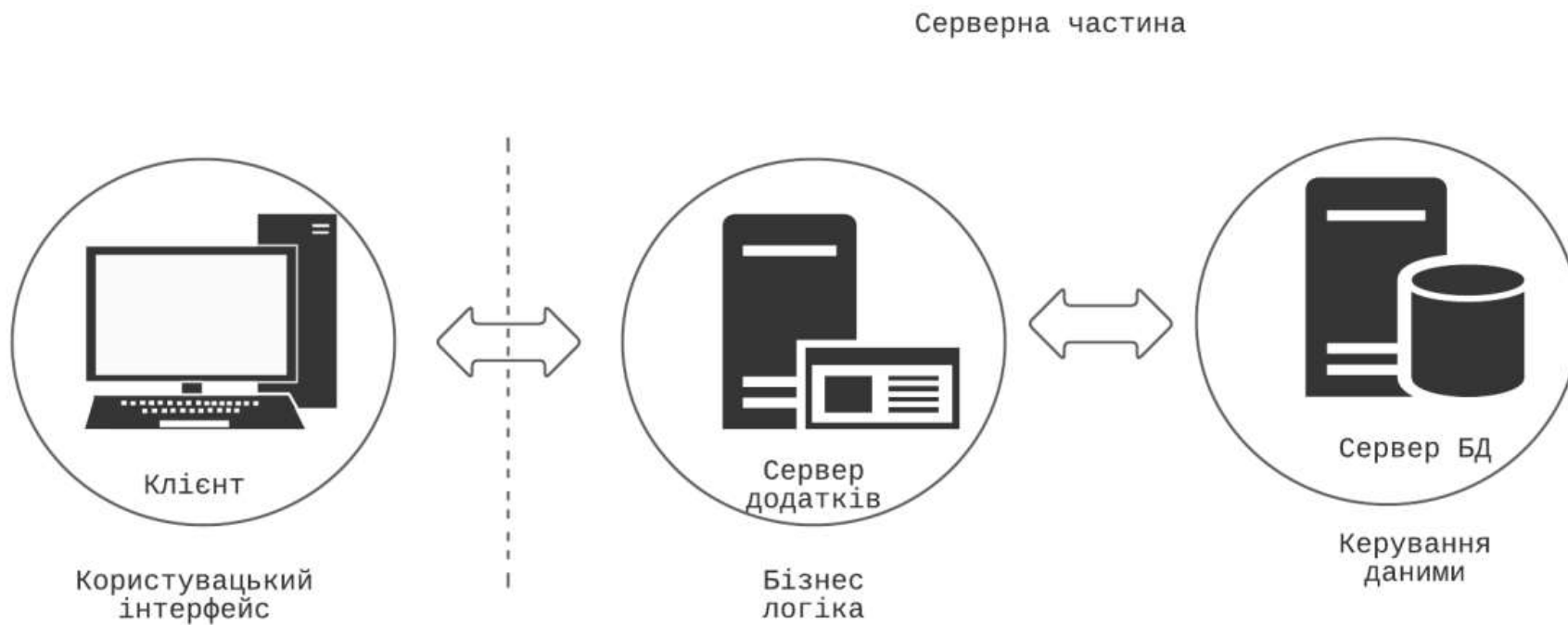
Основні задачі дипломної роботи:

- аналіз існуючих програмних рішень у видавничій сфері;
- формування вимог до нової комп'ютерної системи;
- проєктування архітектури програмного продукту;
- реалізація системи за допомогою сучасних технологій;
- перевірка функціональності та ефективності запропонованого рішення.

Порівняння функціональності сучасних застосунків: Ulysses, Scrivener та Storyist

Функція /Інструмент	Ulysses	Scrivener	Storyist
Текстовий редактор	Distraction-free, Markdown	Розширені опції форматування	Текстовий процесор з форматуванням стилю
Організація проєкту	Групи і підгрупи, фільтри	Коркова дошка, обриси, папки і документи	Картки для персонажів, локацій, сюжетів
Синхронізація	iCloud	Dropbox (також підтримка інших служб)	iCloud
Експорт	PDF, DOCX, ePub, HTML, вебпублікації	PDF, DOCX, ePub, Kindle, RTF та інші	PDF, ePub, RTF, Final Draft
Шаблони документів	Шаблони для блогів та статей	Шаблони для романів, сценаріїв тощо	Шаблони для романів та сценаріїв
Ціль письма	Встановлення та відстеження цілей	Цілі сесій та проєктів	Відстеження прогресу через статистику
Інтерфейс	Мінімалістичний	Комплексний, налаштовуваний	Чистий та організований
Інструменти для рецензії	Коментарі та анотації	Розгорнуті коментарі, анотації, літературне рецензування	Коментарі та перегляд змін
Шаблони	Мінімалістичні та сучасні	Традиційні і гнучкі	Спеціалізовані для письменників і сценаристів
Автоматизація	AppleScript, URL-схеми	AppleScript, компіляція	Недоступна або обмежена
Мобільна версія	iOS	iOS	iOS
Візуалізація даних	Відсутня	Візуалізація даних через карти та обриси	Відсутня

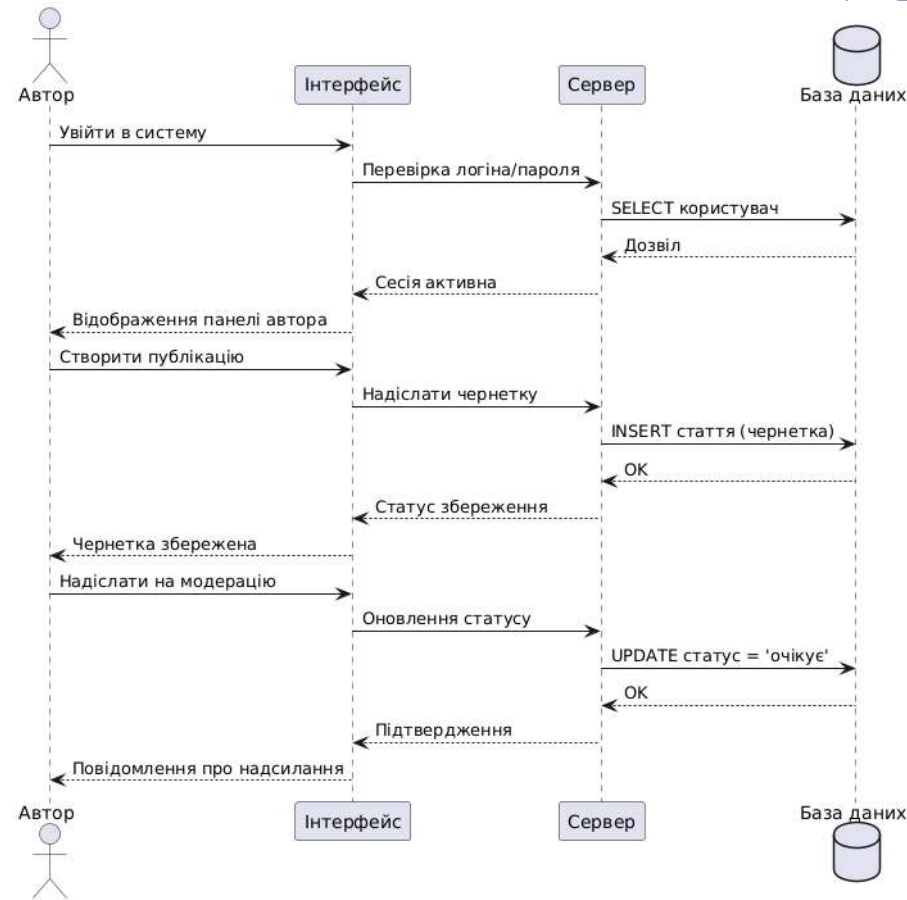
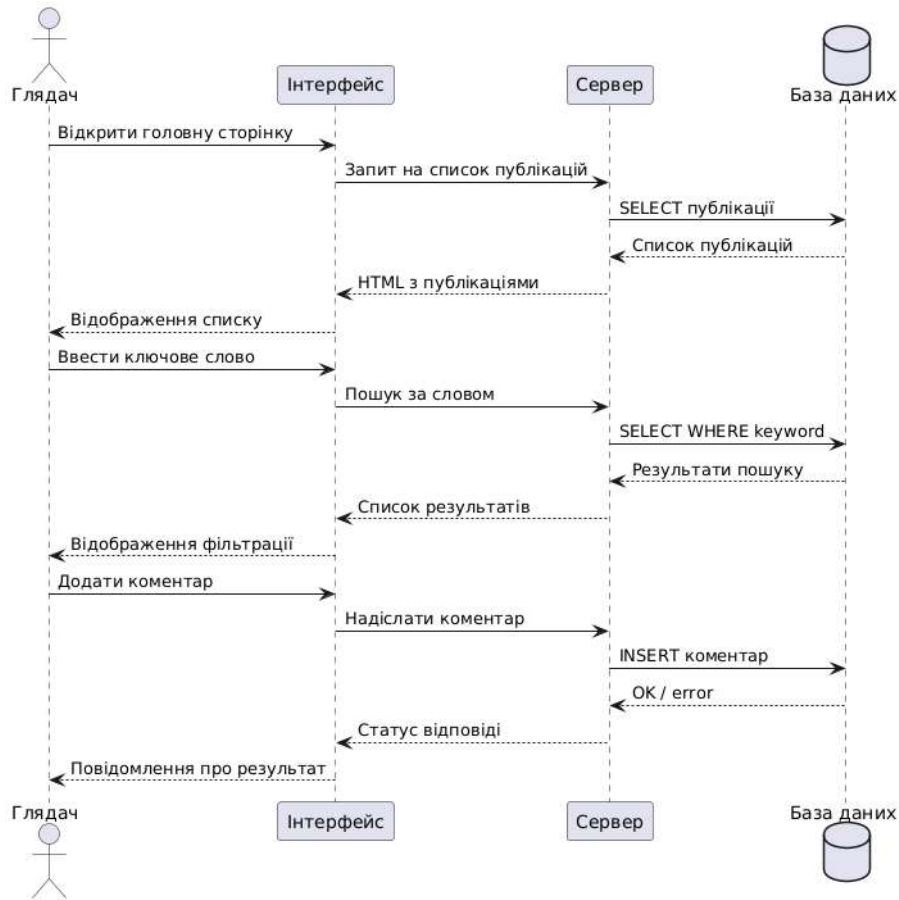
Архітектура вебдодатку



Загальна діаграма варіантів використання видавничої системи



Діаграми послідовності Глядач / Автор



Аналіз технологій розробки

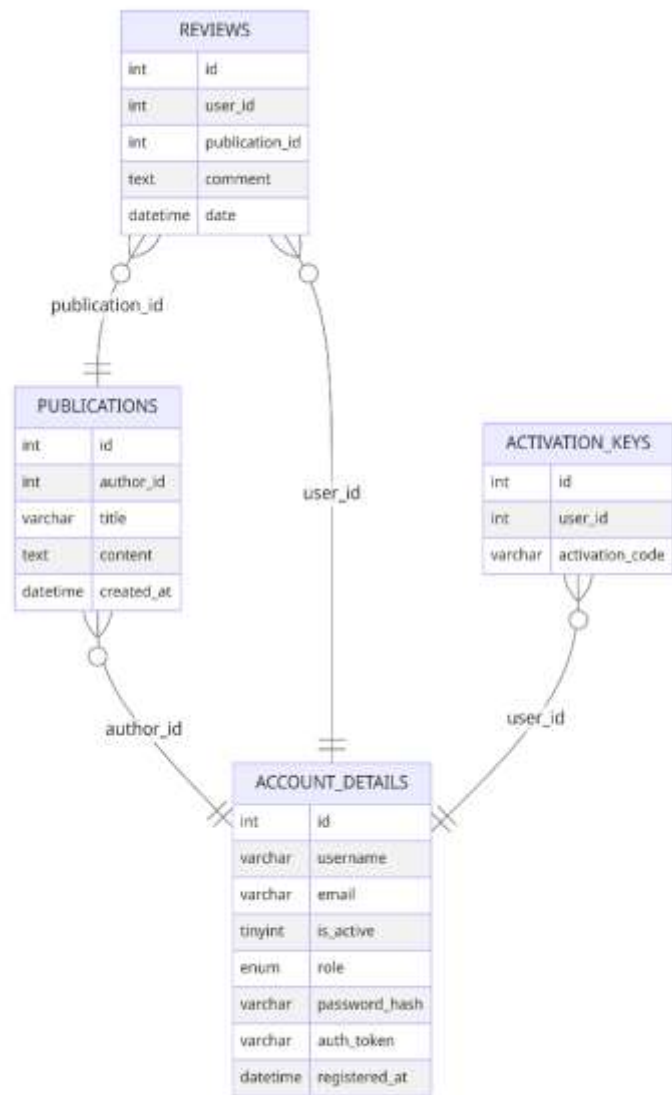
Фреймворк	Продуктивність	Масштабованість	Крива навчання	Екосистема
React	Висока	Висока	Середня	Велика
Angular	Висока	Висока	Висока	Велика
Vue.js	Висока	Середня	Низька	Зростаюча
Bootstrap	Висока	Висока	Середня	Велика

Технологія	Продуктивність	Масштабованість	Екосистема	Крива навчання
Python	Висока	Висока	Велика	Низька
Node.js	Висока	Висока	Велика	Середня
PHP	Висока	Висока	Велика	Низька

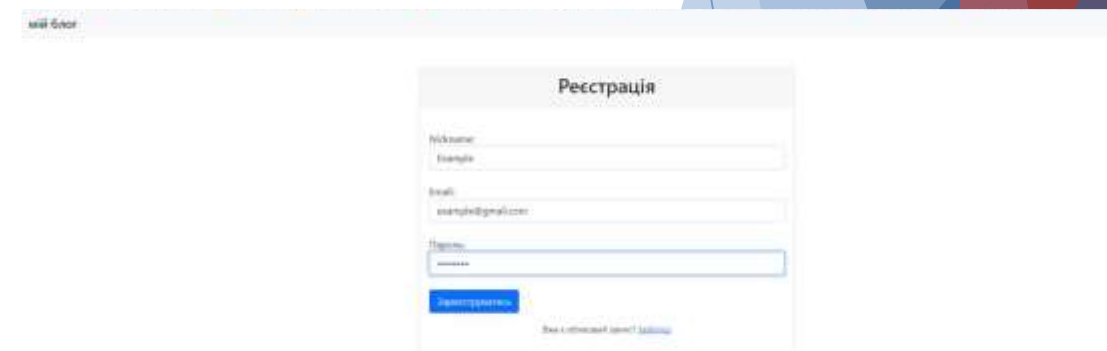
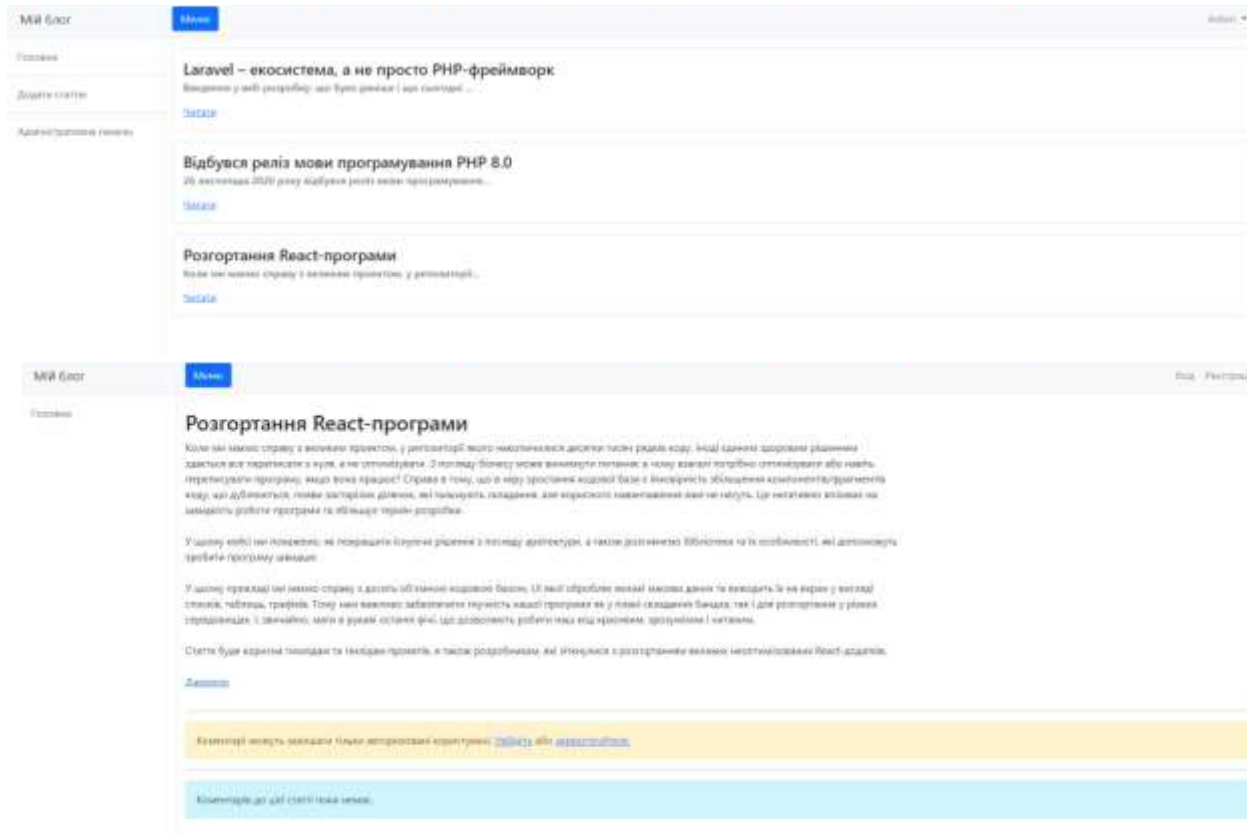
Використані технології

Технологія	Опис
Google Chrome	Вебпереглядач
PhpStorm	Інтегроване середовище розробки
HTML	Мова розмітки
CSS	Мова стилів
Bootstrap	Фреймворк для розробки
PHP	Програмувальна мова
Active Record	Паттерн проєктування
ORM	Паттерн проєктування Object-Relational Mapping
Singleton	Паттерн проєктування
MySQL	Система управління базами даних
phpMyAdmin	Інструмент для роботи з базами даних
Open Server	Локальний сервер
Apache	Вебсервер

Схема бази даних



Інтерфейс користувача



Функціональне тестування

№	Функція	Опис дій	Очікуваний результат	Фактичний результат
1	Створення нової публікації	Увійти в систему як автор; створити нову публікацію з заголовком, текстом та категорією; натиснути "Опублікувати".	Нова публікація створена та відображена на сайті.	Відповідає очікуваному.
2	Редагування публікації	Увійти в систему як автор; внести зміни в існуючу публікацію; натиснути "Зберегти зміни".	Зміни в публікації збережені та відображені.	Відповідає очікуваному.
3	Додавання коментаря	Увійти як користувач; написати коментар до публікації; натиснути "Відправити коментар".	Коментар доданий та відображений під публікацією.	Відповідає очікуваному.
4	Реєстрація нового користувача	Зареєструватися з ім'ям користувача, електронною поштою та паролем; активувати обліковий запис через електронну пошту.	Новий користувач зареєстрований та може увійти.	Відповідає очікуваному.

Висновки

- **Успішна реалізація системи** — розроблено повнофункціональну комп'ютерну систему для видавничої діяльності з підтримкою створення, редагування та публікації текстових матеріалів
- **Комплексний аналіз предметної області** — проведено дослідження існуючих видавничих систем, виявлено їх переваги та недоліки, сформовано вимоги до нового рішення
- **Обґрунтований вибір технологій** — для реалізації використано сучасні веб-технології на клієнті та PHP на сервері з урахуванням вимог безпеки та продуктивності
- **Модульна архітектура** — система має структурований дизайн з окремими компонентами для різних функцій, що забезпечує гнучкість та можливість розширення
- **Успішне тестування** — проведено комплексну перевірку функціональності, продуктивності та безпеки системи, підтверджено відповідність поставленим вимогам
- **Практична придатність** — система готова до впровадження, має зручний користувацький інтерфейс та відповідає критеріям якості
- **Перспективи розвитку** — визначено напрямки для подальшого удосконалення: розширення функціоналу, інтеграція з зовнішніми сервісами, покращення інтерфейсу та автоматизація процесів
- **Досягнення мети роботи** — створено ефективне рішення для підтримки авторів і адміністраторів у процесі організації публікаційного контенту