БАКАЛАВРСЬКА РОБОТА

БР. ІП - 50.00.00.000 ПЗ

Група ІП-21-3

Петруняк Руслан

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Петруняк Руслан Васильович

(прізвище, ім'я, по батькові)

УДК 004

(індекс)

БАКАЛАВРСЬКА РОБОТА

Система автоматизації видавничої діяльності

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього рівня Петруняк Р.В.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Ваврик Тетяна Олександрівна, асистент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Петруняку Руслану Васильовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) **“ Система автоматизації видавничої діяльності ”**

керівник проекту (роботи) Ваврик Т.О. асистент

затверджені наказом закладу вищої освіти від **“** 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Системний аналіз предметної області автоматизації процесів управління у видавничій сфері
2. Специфікація вимог системи автоматизації у видавничій діяльності
3. Розробка архітектури системи автоматизації та управління транзакціями
4. Програмна реалізація системи автоматизації у видавничій діяльності
5. План тестування системи автоматизації у видавничій діяльності

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

1. Діаграма варіантів використання (рис. 2.1)
2. Архітектура MVC (рис. 2.2)
3. ER діаграма (рис. 2.3)
4. Діаграма класів (рис. 2.4)
5. Сторінка входу в систему (рис. 3.1)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Системний аналіз предметної області автоматизації процесів управління у видавничій сфері	04.05.2025	виконано
2	Специфікація вимог системи автоматизації у видавничій діяльності	15.05.2025	виконано
3	Розробка архітектури системи автоматизації та управління транзакціями	21.05.2025	виконано
4	Програмна реалізація системи автоматизації у видавничій діяльності	28.05.2025	виконано
5	План тестування системи автоматизації у видавничій діяльності	03.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 77 сторінок, 38 рисунків, список використаних джерел із 38 найменуваннями.

Мета роботи - розробити програмну систему автоматизації видавничої діяльності з використанням архітектури MVC, яка забезпечує ефективне управління контентом, транзакціями та користувачами.

Об'єкт дослідження - процеси управління видавничою діяльністю у цифровому середовищі.

Предмет дослідження - методи, моделі та програмні засоби автоматизації управління контентом, транзакціями та користувачами у видавничій системі.

В першому розділі проведено системний аналіз предметної області та обґрунтовано потребу в створенні автоматизованої системи для видавничого процесу.

В другому розділі визначено функціональні і нефункціональні вимоги, обґрунтовано архітектуру системи та виконано її логічне проєктування.

В третьому розділі реалізовано ключові модулі системи автоматизації та описано архітектурні компоненти на рівні бекенду.

В четвертому розділі розроблено та проведено тестування системи на відповідність вимогам із використанням сучасних технік генерації тест-кейсів.

Висновок: запропоновано архітектурне рішення автоматизованої системи видавничої діяльності, яке об'єднує сучасні підходи до управління контентом, транзакціями та безпекою в рамках єдиної MVC-архітектури, адаптованої до потреб малих та середніх видавництв.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, ВИДАВНИЧА ДІЯЛЬНІСТЬ, ІНФОРМАЦІЙНА СИСТЕМА, MVC-АРХІТЕКТУРА, УПРАВЛІННЯ КОНТЕНТОМ, ТРАНЗАКЦІЇ, КОРИСТУВАЧІ.

ANNOTATION

The bachelor's thesis contains 77 pages, 38 figures, a list of used sources with 38 names.

The purpose of the work is to develop a software system for automating publishing activities using the MVC architecture, which provides effective management of content, transactions and users.

The object of the study is the processes of managing publishing activities in a digital environment.

The subject of the study is methods, models and software tools for automating content, transactions and users management in the publishing system.

In the first section, a system analysis of the subject area is conducted and the need for creating an automated system for the publishing process is substantiated.

In the second section, functional and non-functional requirements are determined, the system architecture is substantiated and its logical design is performed.

In the third section, the key modules of the automation system are implemented and the architectural components at the backend level are described.

In the fourth section, the system is developed and tested for compliance with the requirements using modern test case generation techniques.

Conclusion: an architectural solution for an automated publishing system is proposed, which combines modern approaches to content management, transactions, and security within a single MVC architecture adapted to the needs of small and medium-sized publishing houses.

KEYWORDS: AUTOMATION, PUBLISHING ACTIVITIES, INFORMATION SYSTEM, MVC ARCHITECTURE, CONTENT MANAGEMENT, TRANSACTIONS, USERS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

SUT – System Under Test – Система, що тестується.

MVC – Model-View-Controller – Модель-Вигляд-Контролер

SPA – Single Page Application – Односторінковий додаток

API – Application Programming Interface – Інтерфейс програмування застосунків

REST – Representational State Transfer – Передача репрезентативного стану

JWT – JSON Web Token – Веб-токен JSON

RBAC – Role-Based Access Control – Контроль доступу на основі ролей

BVA – Boundary Value Analysis – Аналіз граничних значень

EP – Equivalence Partitioning – Розбиття на класи еквівалентності

MTTR – Mean Time To Recovery – Середній час до відновлення

MTTM – Mean Time To Modify – Середній час до модифікації

ВСТУП

Сучасна видавнича сфера активно трансформується під впливом цифрових технологій. Зростаючі вимоги до оперативності, точності, масштабованості та інтеграції з цифровими каналами зумовлюють потребу в автоматизації ключових процесів видавничої діяльності. Автоматизовані системи дозволяють ефективно управляти контентом, обробляти замовлення, здійснювати моніторинг, підтримувати інформаційну безпеку, а також оптимізувати робочі процеси між авторами, редакторами та читачами. Проте в багатьох організаціях і досі спостерігається фрагментарне використання інформаційних технологій або застосування застарілих інструментів, що не відповідають вимогам часу.

У цьому контексті особливої актуальності набуває розробка інтегрованої системи автоматизації видавничої діяльності, яка поєднує функціональність управління контентом, транзакціями, користувачами та аналітичним супроводом на базі сучасної архітектури програмного забезпечення. Обрана архітектура Model-View-Controller (MVC) забезпечує масштабованість, зручність супроводу та розширення системи. Такий підхід дає змогу реалізувати багаторівневу взаємодію між користувачами, розробниками та бізнесом.

Актуальність теми

В умовах цифрової трансформації актуальність створення системи автоматизації видавничої діяльності обумовлена зростанням обсягів інформації, потребою в оперативному оновленні та розповсюдженні контенту, а також високими вимогами до якості управління даними. Більшість видавничих установ стикаються з труднощами при адаптації до цифрових платформ, що призводить до втрати конкурентоспроможності. Використання автоматизованих систем дозволяє вирішити ці проблеми шляхом централізації управління даними, підвищення продуктивності

персоналу та зменшення людського фактору. Запропоноване дослідження дозволяє розробити і впровадити програмну систему, яка відповідає актуальним потребам галузі.

Мета роботи - розробити програмну систему автоматизації видавничої діяльності з використанням архітектури MVC, яка забезпечує ефективне управління контентом, транзакціями та користувачами.

Завдання дослідження

- Провести системний аналіз предметної області автоматизації видавничих процесів.

- Визначити функціональні та нефункціональні вимоги до програмної системи.

- Розробити архітектуру програмного забезпечення на основі моделі MVC.

- Реалізувати модулі системи (аутентифікація, контент, замовлення, користувачі).

- Розробити і виконати план тестування функціональності системи.

- Оцінити ефективність та перспективи практичного впровадження системи.

Об’єкт дослідження - процеси управління видавничою діяльністю у цифровому середовищі.

Предмет дослідження - методи, моделі та програмні засоби автоматизації управління контентом, транзакціями та користувачами у видавничій системі.

Методи дослідження

- Системний аналіз предметної області.

- Метод функціонального моделювання (Use Case, діаграми UML).

- Об’єктно-орієнтоване проектування (діаграми класів, діаграми послідовностей).

- Проєктування на основі шаблонів (MVC-архітектура).

- Методи програмної реалізації на базі сучасних фреймворків.

- Методи тестування (Boundary Value Analysis, Equivalence Partitioning, Use Case Testing).

Наукова новизна

Запропоновано архітектурне рішення автоматизованої системи видавничої діяльності, яке об’єднує сучасні підходи до управління контентом, транзакціями та безпекою в рамках єдиної MVC-архітектури, адаптованої до потреб малих та середніх видавництв.

Практичне значення роботи

Розроблена система може бути впроваджена у реальних видавничих структурах для автоматизації процесів управління контентом, замовленнями та користувачами, що дозволить скоротити витрати часу та ресурсів, підвищити якість обслуговування та забезпечити масштабованість бізнес-процесів.

Бакалаврська робота містить 77 сторінок, 38 рисунків, 4 розділи список використаних джерел із 38 найменуваннями, 1 додаток.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ТА ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ У ВИДАВНИЧІЙ СФЕРІ

1.1. Актуальність розробки системи автоматизації видавничої діяльності

Упродовж останніх років видавнича індустрія зазнала суттєвих трансформацій, зумовлених стрімким розвитком цифрових технологій. Поява електронних книг, цифрових медіа та онлайн-платформ для продажу й розповсюдження призвела до значного ускладнення процесів управління запасами, виконання дистриб'юторських замовлень та ведення точного обліку транзакцій. Ці виклики актуалізують потребу в розробці нового підходу до управління транзакціями, що дозволить оптимізувати операційні процеси, підвищити їхню ефективність та забезпечити цілісність даних.

Традиційно, щоденні видавничі операції, включно з відстеженням запасів, виконанням замовлень та фінансовими транзакціями, здійснювалися вручну. Такі ручні процеси характеризувалися значними витратами часу, високою ймовірністю помилок та низькою ефективністю, що призводило до неточностей в облікових записах. Крім того, вони виявилися неспроможними впоратися з експоненційним зростанням обсягу та складності транзакцій, спричинених цифровою трансформацією галузі.

Для вирішення зазначених проблем були запропоновані різноманітні програмні рішення, такі як системи планування ресурсів підприємства (ERP), бухгалтерське програмне забезпечення та платформи електронної комерції. Однак, ці рішення мають низку обмежень, які роблять їх неоптимальними для задоволення специфічних потреб видавничих компаній. Зокрема, ERP-системи часто є високовартісними та складними для адаптації до унікальних бізнес-процесів видавництв.

Бухгалтерське програмне забезпечення переважно орієнтоване на управління фінансовими потоками і може не містити необхідного функціоналу для ефективного управління товарними запасами та виконання замовлень. З іншого боку, платформи електронної комерції, хоча й ефективні для онлайн-продажів, як правило, не розраховані на обробку складних транзакцій, притаманних видавничій діяльності.

1.2. Функціональні вимоги до системи автоматизації видавничої діяльності

Розробка системи автоматизації видавничої діяльності (САВД) вимагає ретельного визначення її функціональних вимог для забезпечення ефективного управління всіма етапами видавничого процесу. Основними функціональними блоками САВД мають бути:

1. Управління запасами:

- Автоматизоване відстеження наявності та руху друкованої та електронної продукції.

- Формування звітів про рівень запасів, прогнозування потреби у додрукуванні/оновленні цифрових копій.

- Інтеграція з системами складського обліку.

2. Управління замовленнями та дистрибуцією:

- Автоматизована обробка замовлень від дистриб'юторів, роздрібних продавців та кінцевих споживачів.

- Формування пакувальних листів, накладних та інших супровідних документів.

- Відстеження статусу замовлення на всіх етапах його виконання (від оформлення до доставки).

- Можливість інтеграції з логістичними службами.

3. Фінансовий облік та управління транзакціями:

- Автоматизований облік усіх фінансових операцій, пов'язаних з продажем, закупівлею та розподілом.

- Формування рахунків-фактур, актів виконаних робіт, звітів про доходи та витрати.

- Інтеграція з бухгалтерськими системами для забезпечення достовірності фінансових даних.

- Підтримка різних форм оплати та валют.

4 Аналітика та звітність:

- Генерація аналітичних звітів щодо обсягів продажів, прибутковості, динаміки попиту на різні види продукції.

- Візуалізація даних для прийняття управлінських рішень.

- Налаштовувані параметри звітів відповідно до потреб користувача.

5. Управління контентом:

- Зберігання та управління цифровими версіями книг та інших видавничих матеріалів.

- Версіонування контенту, контроль доступу.

1.2.1. Можливі архітектурні рішення для системи автоматизації

При розробці САВД доцільно розглядати архітектурні рішення, які забезпечать масштабованість, гнучкість, надійність та безпеку системи.

1. Мікросервісна архітектура.

Дозволяє розробляти, розгортати та масштабувати окремі функціональні модулі незалежно один від одного. Це спрощує підтримку, оновлення та додавання нового функціоналу без впливу на всю систему.

Кожен з вищезгаданих функціональних блоків (управління запасами, замовленнями, фінансами тощо) може бути реалізований як окремий мікросервіс.

2. Хмарні рішення (Cloud-native)

Використання хмарних платформ (наприклад, AWS, Azure, Google Cloud) забезпечує високу доступність, автоматичне масштабування ресурсів відповідно до навантаження, зниження витрат на інфраструктуру та спрощення розгортання.

САВД може бути розгорнута як SaaS (Software as a Service) рішення, що дозволить видавництвам отримувати доступ до системи через веб-браузер, не турбуючись про інфраструктурні аспекти.

3. API-орієнтований підхід

Надання стандартизованих API (Application Programming Interface) для взаємодії між різними компонентами системи, а також для інтеграції з зовнішніми системами (бухгалтерські програми, платіжні системи, логістичні сервіси).

Дозволить легко інтегрувати САВД з існуючими IT-системами видавництва та розширювати її функціональність за рахунок сторонніх сервісів.

4. Використання реляційних та NoSQL баз даних

Для структурованих даних (інформація про клієнтів, замовлення, фінансові транзакції) доцільно використовувати реляційні СУБД (наприклад, PostgreSQL, MySQL). Для неструктурованих або напівструктурованих даних (журнали подій, контент) можуть бути корисними NoSQL бази даних (наприклад, MongoDB, Cassandra).

Комбінований підхід дозволить оптимально зберігати та обробляти різні типи даних.

5. Принципи безпеки

- Впровадження механізмів автентифікації та авторизації користувачів (наприклад, на основі ролей).

- Шифрування даних як під час передачі, так і під час зберігання.

- Регулярне резервне копіювання даних та плани відновлення після збоїв.

1.3. Призначення та цілі системи управління транзакціями у видавничій діяльності

Метою розробки системи управління транзакціями у видавничій сфері є створення централізованого та автоматизованого інструментарію для оптимізації щоденних операційних процесів видавничого дому. Ця система покликана вирішити наявні проблеми, з якими стикаються видавці під час відстеження запасів, обробки замовлень та забезпечення достовірності облікових записів транзакцій.

1.3.1. Ключові цілі та функціональні можливості

Однією з основних цілей проєкту є стандартизація та цифровізація видавничого процесу. Це дозволить мінімізувати залежність від ручних операцій та паперових записів, які характеризуються низькою ефективністю та високою схильністю до помилок. Система забезпечуватиме доступ до інформації про транзакції в режимі реального часу та автоматизуватиме такі критично важливі завдання, як обробка замовлень та управління запасами. Збір та аналіз даних про продажі, рівні продукції та інші ключові показники дозволить видавцям приймати більш обґрунтовані управлінські рішення та підвищувати загальну ефективність бізнесу.

Додатково, однією з пріоритетних цілей проєкту є підвищення продуктивності та ефективності видавничих процесів. Шляхом оптимізації робочих процесів та автоматизації рутинних операцій, система суттєво скоротить часові та ресурсні витрати на виконання транзакцій.

Це дозволить видавцям перерозподілити зусилля, зосередившись на створенні якісного контенту, замість витрачати значний час на адміністративне управління системою щодо бухгалтерського обліку і фінансових транзакцій.

1.3.2. Безпека даних та адаптивність системи

При розробці системи особлива увага приділяється безпеці даних. Система інтегрує передові методи захисту для забезпечення конфіденційності чутливої інформації, такої як дані транзакцій та банківські реквізити. Це забезпечить видавцям впевненість у захисті своїх даних та мінімізує ризики витоків інформації та інших загроз безпеці.

Завдяки використанню сучасних інструментів веброзробки, система демонструє високу гнучкість та адаптивність. Її архітектура дозволяє модифікувати та масштабувати систему відповідно до унікальних потреб різних видавничих будинків, враховуючи їхній розмір, обсяг діяльності та бізнес-модель. Інтуїтивно зрозумілий та дружній інтерфейс користувача сприяє швидкому освоєнню системи, ефективному пошуку інформації та оперативному виконанню транзакцій.

Обсяг проєкту охоплює комплексне управління транзакціями, починаючи від управління продукцією і закінчуючи фінансовими операціями. Основне призначення системи полягає у наданні уніфікованого інструментарію для управління транзакціями, автоматизації процесів, підвищення операційної ефективності та гарантування безпеки даних. Гнучкість методології розробки дозволяє адаптувати систему до специфічних потреб різних видавничих суб'єктів. Загалом, проєкт націлений на надання видавцям повного та сучасного рішення, адаптованого до викликів та можливостей цифрової ери.

1.4. Організація проєкту розробки системи управління транзакціями у видавничій діяльності

Ефективне управління проєктом є критично важливим для його своєчасного та якісного завершення. Проєкт розробки системи управління транзакціями у видавничій сфері передбачає послідовне проходження таких

етапів: дослідження, планування, виконання, тестування та запуск. З метою забезпечення контролю та відповідності графіку буде розроблено детальний план проєкту, що включатиме перелік завдань, часові рамки та ключові етапи. Крім того, буде визначено список артефактів проєкту, а саме: план проєкту, специфікації вимог, дизайнерські документи, вихідний код та результати тестування. Для контролю версій вихідного коду та відстеження змін у процесі розробки буде використана система контролю версій. Керівник проєкту отримуватиме регулярні звіти про прогрес для моніторингу виконання плану та оперативного вирішення будь-яких виникаючих проблем.

1.4.1. Модель життєвого циклу проєкту

Для реалізації проєкту буде застосовано Agile-методологію, яка є ітеративним підходом, що акцентує увагу на співпраці між членами команди та зацікавленими сторонами, а також на гнучкості та адаптивності до змін. Ця модель передбачає декомпозицію проєкту на менші, керовані ітерації, відомі як спринти. Кожен спринт завершується створенням функціонального продукту, який підлягає перегляду та модифікації на основі отриманого зворотного зв'язку. Такий підхід забезпечує відповідність кінцевого продукту потребам користувачів, сприяє більш частій та ефективній комунікації із зацікавленими сторонами. Застосування Agile-методології також дозволяє швидко реагувати на зміни вимог та мінімізувати ризики шляхом раннього виявлення потенційних проблем. Agile базується на принципах безперервної інтеграції та безперервної поставки. Це означає, що тестування та інтеграція компонентів програмного забезпечення здійснюються ітеративно протягом усього проєкту, а не лише на завершальному етапі. Цей підхід прискорює цикл зворотного зв'язку, спрощує ідентифікацію та вирішення проблем, а також забезпечує ефективніше управління ресурсами.

1.4.2. Аналіз ризиків

Аналіз ризиків є інтегральною частиною плану управління проєктом. Він передбачає ідентифікацію потенційних ризиків, що можуть перешкодити успішному завершенню проєкту, оцінку ймовірності їх виникнення та потенційної серйозності, а також розробку стратегій їхнього зниження. У контексті розробки програмного забезпечення були виявлені такі технічні ризики: програмні дефекти, проблеми з апаратним забезпеченням та питання сумісності платформ. До ризиків управління проєктом належать: затримки у графіку, розширення обсягу проєкту та недостатня підтримка зацікавлених сторін.

Стратегія управління ризиками включає розробку планів резервного копіювання для найбільш критичних ризиків. Наприклад, у разі виникнення непередбачених технічних проблем, що можуть призвести до затримок у графіку, передбачено резервний план, який передбачає потенційне скорочення обсягу проєкту для досягнення цілей. Інший план резервного копіювання передбачає залучення додаткових ресурсів до проєкту, якщо буде виявлено відставання команди від графіка.

1.4.3. Вимоги до апаратного забезпечення

Для забезпечення оптимальної продуктивності системи визначено наступні мінімальні вимоги до апаратного забезпечення:

- Процесор: Intel Core i5 або еквівалентний (вищого покоління)

- Оперативна пам'ять: 8 ГБ або більше

- Накопичувач: Твердотільний накопичувач (SSD) об'ємом не менше 256 ГБ

- Мережа: Підключення до Інтернету через Ethernet або Wi-Fi

- Дисплей: Монітор з діагоналлю 15 дюймів і роздільною здатністю не менше 1920x1080 пікселів

Для ефективного моніторингу та своєчасного завершення проєкту визначено основні поставки та розроблено графік проєкту. Ключові поставки включають:

- Детальний документ специфікації системних вимог
- Комплексний документ архітектурного дизайну
- Повністю функціональна та протестована система
- Керівництво користувача для системи
- Звіт про завершення проєкту

Графік проєкту структурований на кілька етапів: збір вимог, дизайн, розробка, тестування та впровадження. Кожен етап має визначені терміни виконання та перелік дій. Регулярні контрольні точки та огляди передбачені для забезпечення відповідності ходу проєкту плану. Для ефективного моніторингу прогресу та управління графіком буде використовуватися програмне забезпечення для управління проєктами, що забезпечує відстеження завдань та оновлення статусу в режимі реального часу.

1.4.4. Механізми моніторингу, звітності та контролю

Для забезпечення контролю та своєчасного виявлення й усунення відхилень від плану, в проєкті будуть впроваджені механізми моніторингу, звітності та контролю. Це передбачає регулярний моніторинг та звітність щодо прогресу проєкту стосовно графіка, бюджету та обсягу. Системи контролю забезпечуватимуть коригувальні дії для виправлення будь-яких відхилень від плану. Для моніторингу процесів, генерації звітів та їх обробки можуть бути використані спеціалізовані інструменти, такі як Microsoft Project. Крім того, проведення регулярних нарад щодо статусу проєкту та інформування зацікавлених сторін про прогрес проєкту сприятиме підтримці обізнаності та оперативному вирішенню виникаючих проблем. Завдяки впровадженню цих процесів буде забезпечено відповідність проєкту плану та ефективне вирішення будь-яких відхилень.

1.4.5. Вплив проєкту на окремих осіб та організації

Проєкт розробки системи управління транзакціями матиме значний позитивний вплив як на індивідуальних користувачів, так і на організації. Очікується, що система прискорить обробку транзакцій, підвищить точність та ефективність фінансових операцій. Це дозволить окремим особам та організаціям перерозподілити фокус на інші критично важливі завдання, що сприятиме підвищенню продуктивності та прибутковості. Крім того, система забезпечить точні та надійні фінансові записи, мінімізуючи потенційні помилки під час обробки транзакцій.

Впровадження цієї системи також створить нові можливості працевлаштування для фахівців, відповідальних за її підтримку та обслуговування. В цілому, проєкт принесе користь суспільству шляхом підвищення прозорості фінансових операцій та забезпечення ефективного моніторингу фінансової діяльності з боку державних органів. Таким чином, впровадження Системи управління транзакціями матиме позитивний вплив на окремих осіб, організації та суспільство в цілому.

РОЗДІЛ 2. СПЕЦИФІКАЦІЯ ВИМОГ ТА РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ АВТОМАТИЗАЦІЇ ТА УПРАВЛІННЯ ТРАНЗАКЦІЯМИ У ВИДАВНИЧІЙ ДІЯЛЬНОСТІ

2.1. Ідентифікація стейкхолдерів Системи автоматизації видавничої діяльності

У контексті розробки Системи автоматизації видавничої діяльності (САВД), стейкхолдери визначаються як особи, групи або організації, що мають прямий чи опосередкований вплив на систему або зацікавлені в її функціонуванні та результатах. Для даного проєкту були ідентифіковані наступні ключові групи стейкхолдерів:

1. Автори та творці контенту.

Ця категорія стейкхолдерів є первинними генераторами інтелектуального продукту. Вони використовуватимуть систему для створення, завантаження та, потенційно, публікації своїх рукописів, ілюстрацій чи інших медіа-матеріалів. Їхні потреби стосуються зручності інтерфейсу, функціоналу для завантаження файлів, відстеження статусу матеріалів та, можливо, інструментів для співавторства.

2. Редактори та рецензенти.

Представники цієї групи відповідають за якість та відповідність контенту редакційним стандартам. Вони використовуватимуть систему для перегляду, редагування, рецензування та внесення правок до матеріалів, поданих авторами. Для них важлива функціональність коментування, відстеження змін, інструменти для спільної роботи та можливість керування редакційним процесом.

3. Дистриб'ютори та видавці.

Ці стейкхолдери є основними користувачами системи з точки зору комерційної реалізації контенту. Вони використовуватимуть САВД для

управління процесами розповсюдження, продажу та обліку видавничої продукції (як друкованої, так і електронної). Їхні потреби включають функціонал управління замовленнями, інвентаризацію запасів, формування звітів про продажі, інтеграцію з платіжними системами та платформами електронної комерції.

4. Адміністратори та модератори.

Ця група стейкхолдерів відповідає за підтримку та належне функціонування системи. Вони використовуватимуть систему для управління обліковими записами користувачів, модерації контенту, налаштування системних параметрів, моніторингу продуктивності та виконання інших адміністративних завдань. Для них важливий доступ до інструментів адміністрування, системних журналів, можливості управління правами доступу та конфігурацією.

Визначення та врахування потреб кожної групи стейкхолдерів є ключовим для успішної розробки та впровадження САВД, оскільки це забезпечить її релевантність, функціональність та прийняття кінцевими користувачами.

2.2. Діаграма варіантів використання та специфікація функціоналу системи

На рисунку 2.1 представлена діаграма варіантів використання, що ілюструє взаємодію основних стейкхолдерів із системою управління транзакціями у видавничій діяльності. Ця діаграма відображає функціональні можливості системи на високому рівні, розкриваючи ролі та дії кожного типу користувачів.

Наведемо опис ролей та їх функціоналу.

- Автори. Ця роль передбачає функціонал зі створення та управління власними публікаціями. Автори мають можливість ініціювати нові

публікації, створювати та редагувати окремі розділи в межах своїх робіт. Крім того, їм доступна функція редагування інформації власного профілю.

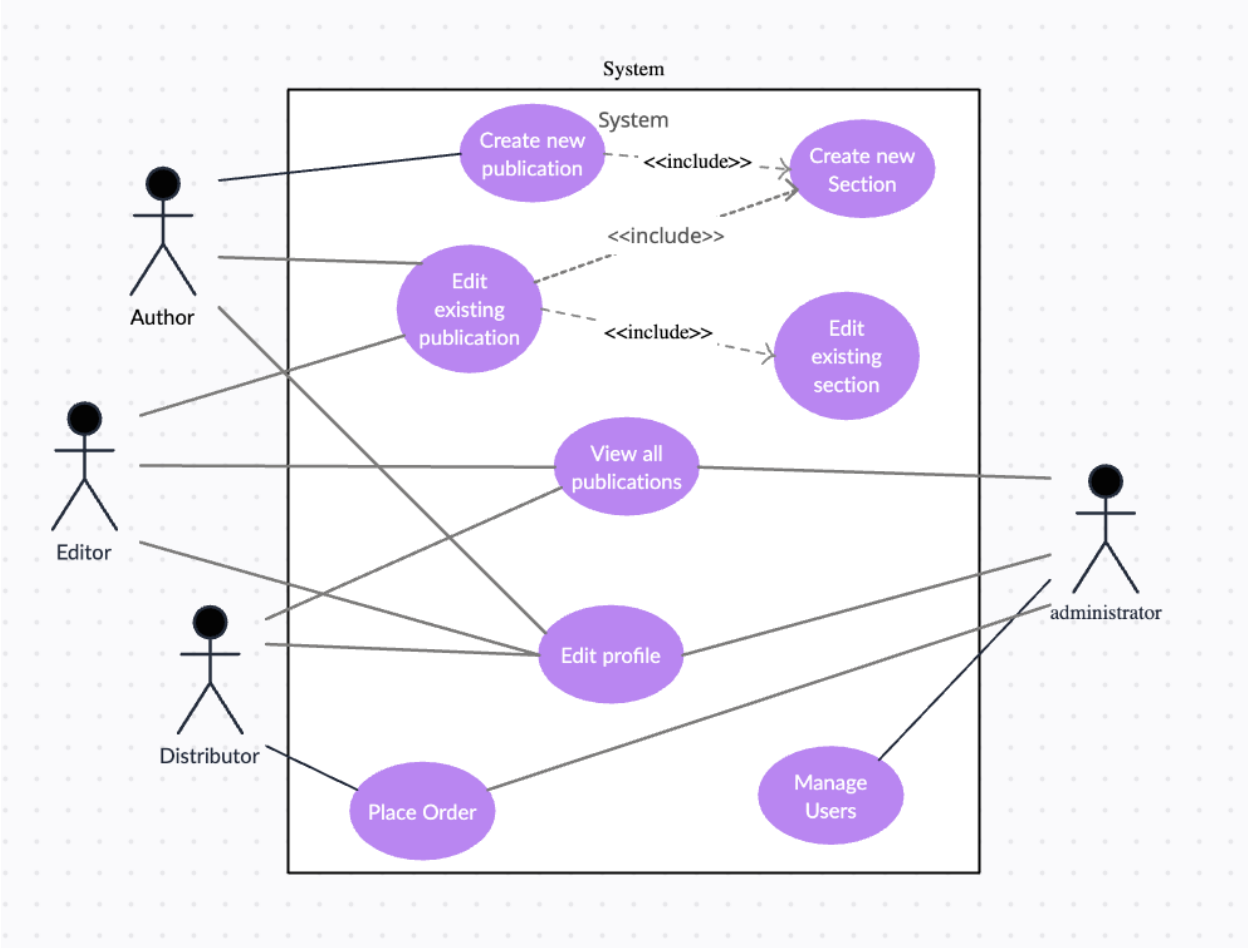

Рисунок 2.1 - Діаграма варіантів використання

2. Редактори. Редактори здійснюють редагування публікацій, створених авторами. Їх функціонал включає перегляд всіх розділів публікації, створення нових розділів та модифікацію існуючих.

3. Дистриб'ютори. Дистриб'ютори мають можливість переглядати всі публікації, доступні в системі, та розміщувати замовлення на друк та розповсюдження обраних публікацій.

4. Адміністратори. Адміністратори системи відповідають за управління всіма користувачами, зареєстрованими в системі. Окрім специфічних

адміністративних функцій, адміністратор може виконувати всі операції, доступні іншим ролям.

2.2.1. Детальна специфікація варіантів використання

Нижче наведено детальний опис кожного варіанту використання, представленого на діаграмі:

1. Створення нової публікації.

Цей варіант використання дозволяє автору додавати нові публікації (наприклад, книги, журнали) до бази даних системи. При створенні нової публікації автор вводить обов'язкову інформацію, що включає назву, підзаголовок, категорію, тематику та ціну.

2. Редагування існуючої публікації.

Функціонал надає можливість автору або редактору модифікувати існуючу публікацію. Після вибору публікації для редагування відображається інтерфейс, що дозволяє внести зміни до її атрибутів. Модифікація публікації може також включати додавання нових розділів або внесення змін до вже існуючих.

3. Створення нового розділу.

Цей варіант використання дозволяє автору додавати нові розділи до наявної публікації. При створенні нового розділу автор надає необхідну інформацію, таку як тип розділу, назва та вміст.

4. Редагування існуючого розділу.

Цей функціонал дозволяє автору або редактору модифікувати вже існуючий розділ публікації. При виборі розділу для редагування відображається його поточний вміст, який може бути змінений та збережений.

5. Перегляд усіх публікацій.

Даний варіант використання забезпечує доступ до переліку всіх публікацій, що знаходяться в системі. Цей модуль доступний для всіх

стейкхолдерів з їхньої панелі управління та відображає інформацію про кожну публікацію, включаючи дані про автора.

6. Розміщення замовлення.

Цей функціонал дозволяє дистриб'ютору ініціювати замовлення на будь-яку з публікацій, що доступні для друку та розповсюдження. В процесі розміщення замовлення система автоматично розраховує та відображає загальну вартість.

7. Управління користувачами.

Цей варіант використання є виключною прерогативою адміністратора системи. Він дозволяє адміністратору переглядати, редагувати деталі та інформацію про ролі всіх зареєстрованих користувачів.

2.3. Нефункціональні вимоги до системи автоматизації та управління транзакціями

Окрім функціональних вимог, для забезпечення ефективності, надійності та стійкості системи управління транзакціями необхідно визначити низку нефункціональних вимог. Ці вимоги стосуються якості та експлуатаційних характеристик системи, а не її специфічних функцій.

1. Продуктивність: Система повинна демонструвати високу швидкість відгуку, забезпечуючи час реакції на запити користувачів не більше ніж дві секунди. Це критично для забезпечення безперебійної роботи користувачів.

2. Безпека: Система автоматизації повинна гарантувати безпеку всіх транзакцій та конфіденційність даних. Це включає впровадження надійних механізмів автентифікації користувачів та застосування методів шифрування приватних даних для запобігання несанкціонованому доступу до інформації.

3. Масштабованість: Система повинна бути масштабованою, тобто здатною ефективно обробляти зростаючий обсяг користувачів та транзакцій

без деградації продуктивності. Це забезпечить стабільну роботу при збільшенні навантаження на систему.

4. Надійність: Система повинна характеризуватися високою надійністю, забезпечуючи доступність системи на рівні 99% часу. Максимально допустимий місячний час простою для обслуговування не повинен перевищувати однієї години.

5. Зручність використання (Usability): Користувацький інтерфейс системи повинен бути інтуїтивно зрозумілим та мати очевидну навігацію. Кількість кроків, необхідних для виконання транзакції, має бути мінімізованою для забезпечення високої ефективності взаємодії користувача.

6. Сумісність: Система повинна підтримувати сумісність з різними веб-браузерами, операційними системами та іншими програмними додатками, що використовуються в інфраструктурі компанії. Це забезпечить гнучкість впровадження та використання СУТ.

7. Підтримуваність (Maintainability): СУТ повинна бути легкою для оновлення та виправлення помилок. Це досягається за рахунок чіткої та всеосяжної документації, а також застосування модульної архітектури, що спрощує модифікацію та розширення функціоналу.

8. Цілісність даних: Система повинна гарантувати цілісність даних, забезпечуючи коректний запис та зберігання кожної транзакції. Для цього мають бути передбачені процедури резервного копіювання та відновлення даних для запобігання втратам.

2.4. Метрики вимірювання відповідності нефункціональним вимогам

Для об'єктивної оцінки та верифікації відповідності розроблюваної системи визначеним нефункціональним вимогам, необхідно використовувати конкретні, вимірювані метрики.

1. Продуктивність

- Час відгуку (Response Time):

- Метрика: Середній та максимальний час, необхідний системі для обробки запиту користувача та повернення результату.

- Ціль: ≤ 2 секунди для 95% запитів.

- Вимірювання: Використання інструментів моніторингу продуктивності (наприклад, Apache JMeter, LoadRunner) для симуляції навантаження та вимірювання часу відгуку для ключових операцій (наприклад, створення публікації, розміщення замовлення, перегляд списку публікацій).

- Пропускна здатність (Throughput):

- Метрика: Кількість транзакцій або запитів, які система може обробити за одиницю часу.

- Ціль: Не менше X транзакцій за секунду (де X визначається на основі очікуваного пікового навантаження).

- Вимірювання: Тестування навантаження та стрес-тестування для визначення максимальної пропускної здатності до деградації продуктивності.

2. Безпека

- Частота успішних автентифікацій (Authentication Success Rate):

- Метрика: Відсоток успішних спроб автентифікації користувачів.

- Ціль: 100% успішних автентифікацій при коректних облікових даних.

- Вимірювання: Моніторинг системних логів автентифікації.

- Відсоток вразливостей (Vulnerability Percentage):

- Метрика: Кількість виявлених критичних/високих вразливостей після проведення сканування безпеки та пентесту.

- Ціль: 0 критичних та високих вразливостей.

- Вимірювання: Регулярне автоматизоване сканування безпеки (SAST/DAST) та ручне тестування на проникнення (пентест).

- Відповідність стандартам безпеки (Compliance with Security Standards):

- Метрика: Наявність та відповідність впроваджених заходів безпеки галузевим стандартам (наприклад, OWASP Top 10, ISO/IEC 27001).

- Ціль: Повна відповідність визначеним стандартам та рекомендаціям.

- Вимірювання: Аудит безпеки та перевірка документації.

3. Масштабованість

- Час масштабування (Scaling Time):

- Метрика: Час, необхідний для автоматичного або ручного збільшення/зменшення ресурсів системи для обробки зростаючого/зменшуючого навантаження.

- Ціль: До Y хвилин для горизонтального масштабування.

- Вимірювання: Проведення тестів навантаження зі змінним трафіком та фіксація часу реакції системи на зміну навантаження.

- Вартість за транзакцію/користувача при масштабуванні (Cost per Transaction/User at Scale):

- Метрика: Зміна операційних витрат на одиницю транзакції або на одного користувача при збільшенні кількості користувачів/навантаження.

- Ціль: Збереження лінійного або суб-лінійного зростання витрат.

- Вимірювання: Аналіз витрат на хмарні ресурси або апаратне забезпечення при різних рівнях навантаження.

4. Надійність

- Час безвідмовної роботи (Uptime):

- Метрика: Відсоток часу, протягом якого система доступна та функціонує належним чином.

- Ціль: $\geq 99\%$ (що відповідає приблизно 7 годинам і 12 хвилинам простою на місяць).

- Вимірювання: Постійний моніторинг доступності системи за допомогою зовнішніх інструментів (наприклад, UptimeRobot, Prometheus).

- Середній час до відновлення (Mean Time To Recovery, MTTR):

- Метрика: Середній час, необхідний для відновлення системи після збою.

- Ціль: ≤ 30 хвилин.

- Вимірювання: Фіксація часу відновлення після симульованих збоїв або реальних інцидентів.

5. Зручність використання (Usability)

- Час виконання типового завдання (Time to Complete a Typical Task):

- Метрика: Середній час, необхідний користувачеві для виконання типового завдання (наприклад, створення нової публікації, розміщення замовлення).

- Ціль: ≤ 5 хвилин для складних завдань, ≤ 30 секунд для простих.

- Вимірювання: Користувацьке тестування (usability testing) із залученням цільової аудиторії, фіксація часу виконання завдань.

- Рівень помилок користувача (User Error Rate):

- Метрика: Кількість помилок, допущених користувачами під час взаємодії з системою за певний період або під час виконання певного завдання.

- Ціль: $\leq 2\%$ помилок.

- Вимірювання: Аналіз логів користувацької взаємодії, спостереження під час користувацького тестування.

- Показник задоволеності користувачів (User Satisfaction Score):

- Метрика: Оцінка задоволеності користувачів системою, отримана через опитування (наприклад, System Usability Scale - SUS).

- Ціль: SUS score ≥ 70.

- Вимірювання: Проведення опитувань після впровадження та періодично.

6. Сумісність

- Відсоток успішних тестів сумісності (Compatibility Test Success Rate):

- Метрика: Частка успішно пройдених тестів на сумісність з різними веб-браузерами (Chrome, Firefox, Edge), операційними системами (Windows, macOS, Linux) та ключовими корпоративними додатками.

- Ціль: 100% успішних тестів для визначеного набору конфігурацій.

- Вимірювання: Проведення крос-браузерного та крос-платформенного тестування.

- Час інтеграції з зовнішніми системами (Integration Time with External Systems):

- Метрика: Час, необхідний для успішної інтеграції СУТ з іншими критично важливими корпоративними системами (наприклад, ERP, бухгалтерське ПЗ).

- Ціль: $\leq X$ днів/тижнів на інтеграцію (де X залежить від складності інтеграції).

- Вимірювання: Фіксація часу, витраченого на розробку та тестування інтеграційних модулів.

7. Підтримуваність

- Середній час до модифікації (Mean Time To Modify, MTTM):

- Метрика: Середній час, необхідний для реалізації зміни або додавання нового функціоналу (наприклад, нового типу звіту).

- Ціль: $\leq X$ днів/тижнів.

- Вимірювання: Відстеження часу, витраченого розробниками на реалізацію запитів на зміни.

- Індекс читабельності коду (Code Readability Index):

- Метрика: Оцінка читабельності та зрозумілості вихідного коду за допомогою інструментів статичного аналізу коду.

- Ціль: Досягнення визначеного порогу (наприклад, за метрикою цикломатичної складності, або певного балу в інструментах, що оцінюють чистоту коду).

- Вимірювання: Автоматичний аналіз коду за допомогою SonarQube, ESLint тощо.

8. Цілісність даних

- Відсоток втрати даних під час збою (Data Loss Percentage During Failure):

- Метрика: Обсяг даних, втрачених у випадку системного збою.

- Ціль: 0%.

- Вимірювання: Проведення тестування відмовостійкості та відновлення даних після симульованих збоїв.

- Час відновлення даних (Data Recovery Time):

- Метрика: Час, необхідний для повного відновлення даних з резервної копії після критичного збою.

- Ціль: $\leq Y$ годин.

- Вимірювання: Практичні випробування процедур резервного копіювання та відновлення.

- Відсоток успішних транзакцій (Successful Transaction Rate):

- Метрика: Частка транзакцій, які були успішно записані та збережені без помилок.

- Ціль: 100%.

- Вимірювання: Моніторинг логів транзакцій та перевірка цілісності бази даних.

Використання цих метрик дозволить не тільки контролювати процес розробки, але й забезпечити відповідність кінцевого продукту високим стандартам якості та вимогам бізнесу.

2.5. Архітектура Model-View-Controller (MVC) у розробці системи автоматизації

Архітектура Model-View-Controller (MVC) є поширеним шаблоном проєктування програмного забезпечення, що забезпечує логічне розділення компонентів програми на три основні взаємопов'язані частини: Модель (Model), Вигляд (View) та Контролер (Controller). Цей підхід сприяє підвищенню модульності, гнучкості та підтримуваності програмних систем.

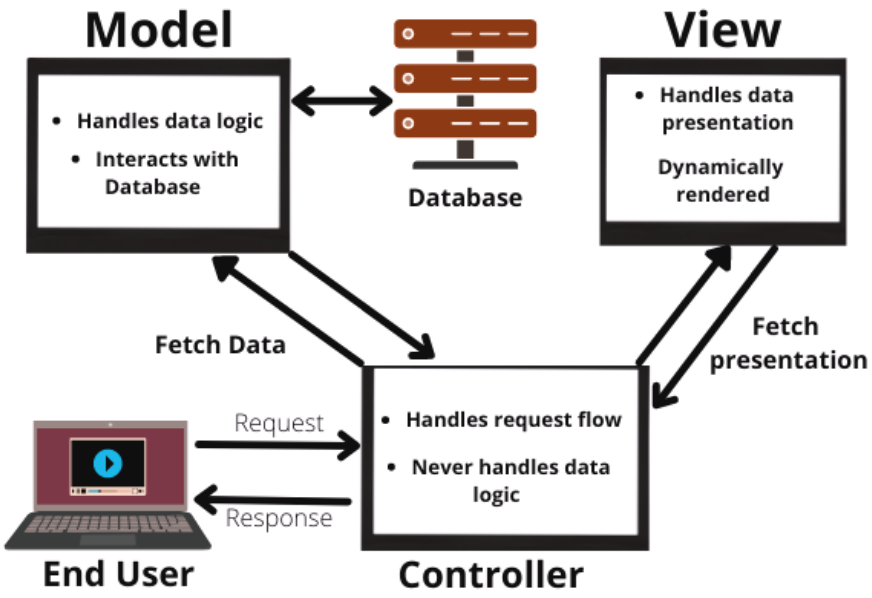

Рисунок 2.2 – Архітектура MVC

Розглянемо принципи компонентів MVC.

1. Контролер (Controller).

Цей компонент відповідає за обробку введення від користувача. Він інтерпретує дії користувача, взаємодіє з Моделлю для оновлення даних та з Виглядом для відображення відповідних змін. Контролер виступає як посередник, який координує взаємодію між Моделлю та Виглядом.

2. Вигляд (View).

Вигляд відповідає за представлення даних користувачеві у зрозумілій та інтерактивній формі. Його основна функція полягає у візуалізації

інформації, отриманої від Моделі, і відображенні її кінцевому користувачеві. Вигляд не містить бізнес-логіки.

3. Модель (Model).

Модель інкапсулює дані програми та бізнес-логіку. Вона відповідає за зберігання, управління та обробку даних, а також за забезпечення їхньої цілісності та узгодженості. Модель не взаємодіє безпосередньо з користувацьким інтерфейсом.

2.5.1. Застосування архітектури MVC у пропонованій системі

У контексті розробки системи автоматизації та управління транзакціями, компоненти MVC можуть бути реалізовані наступним чином:

- Модель - представлятиме базу даних та бізнес-логіку, що використовуються для обробки транзакцій, управління запасами та замовленнями. Це включатиме реалізацію алгоритмів для обчислень, механізмів забезпечення цілісності даних, а також функцій для додавання, зміни та видалення записів у базі даних.

- Вигляд - відповідатиме за відображення інформації користувачеві, такої як записи транзакцій, статус замовлень та поточні обсяги товарів на складі, у доступній та зручній формі.

- Контролер - оброблятиме введення користувача (наприклад, додавання нових транзакцій, зміну замовлень, відстеження кількості товарів) та координуватиме взаємодію з Моделлю та Виглядом для виконання необхідних операцій.

2.5.2. Переваги архітектури MVC для системи автоматизації

Застосування архітектури MVC для розробки є доцільним з кількох причин:

1. Чітке розділення відповідальності. MVC забезпечує високий ступінь розділення відповідальності між компонентами програми. Це значно

спрощує підтримку та розширення кожної частини незалежно, що призводить до зменшення кількості помилок та підвищення ефективності розробки. Така модульність також сприяє більшій гнучкості та адаптивності, оскільки окремі компоненти можуть бути вдосконалені або замінені без суттєвого впливу на інші частини системи.

2. Підтримка веб-додатків та покращений користувацький досвід. Дизайн MVC ідеально підходить для веб-додатків, оскільки він полегшує розділення серверного та клієнтського коду. Типово, Вигляд реалізується на клієнтській стороні (за допомогою HTML, CSS та JavaScript), тоді як Модель та Контролер розгортаються на серверній стороні. Це дозволяє клієнтському коду обробляти взаємодії та оновлення інтерфейсу без необхідності перезавантаження всієї сторінки, що забезпечує швидший та більш динамічний користувацький досвід.

2.6. Архітектурна модель системи автоматизації та управління транзакціями на основі MVC

Представлена архітектурна модель для розроблюваної системи базується на патерні Model-View-Controller (MVC), що забезпечує чітке розділення відповідальності між компонентами системи.

Компонент Модель відповідає за управління даними та бізнес-логікою. У системі він включає наступні сутності:

- Модель даних публікації: Містить атрибути, пов'язані з книгами та іншими видавничими продуктами, такі як ідентифікатор публікації (ID), назва, автор, жанр, ціна та кількість (інвентарний запас).

- Модель даних замовлення: Інкапсулює інформацію про транзакції, включаючи ідентифікатор транзакції (ID), дату, ідентифікатор публікації (ID книги), кількість замовлених одиниць та загальну вартість.

Модель даних користувача: Зберігає дані, пов'язані з користувачами системи, а саме: ідентифікатор користувача (ID), ім'я, адресу електронної пошти та хешований пароль.

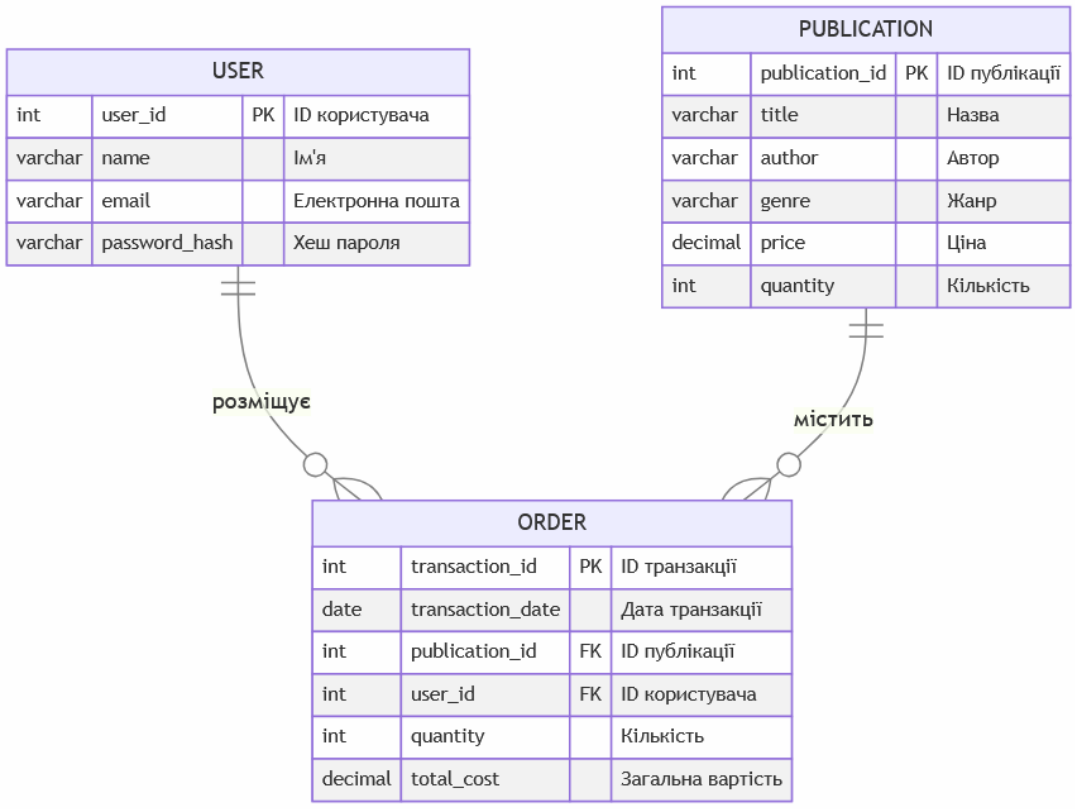

Рисунок 2.3 – ER діаграма

Компонент Вигляд (View) відповідає за представлення даних користувачеві. У СУТ він реалізований через такі інтерфейси:

- Вигляд публікацій: Відображає список доступних публікацій в інвентарі, надаючи функціонал для додавання, редагування або видалення позицій.

- Вигляд замовлень: Представляє перелік здійснених замовлень з можливістю їх фільтрації за датою або ідентифікатором публікації.

- Вигляд користувача: Надає форми для автентифікації (входу) та реєстрації нових користувачів у системі.

Компонент Контролер (Controller) обробляє користувацькі запити, взаємодіє з Моделлю для виконання операцій та оновлює Вигляд. У СУТ функціонал Контролера розподілений таким чином:

- Контролер публікацій: Обробляє запити, пов'язані з управлінням інвентарем, такі як додавання, редагування або видалення публікацій зі складу.

- Контролер замовлень: Опрацьовує запити, що стосуються управління замовленнями, включаючи створення нових транзакцій або фільтрацію існуючих за датою чи ідентифікатором публікації.

- Контролер користувача: Відповідає за обробку запитів, пов'язаних з автентифікацією та реєстрацією користувачів, зокрема перевірку облікових даних та створення нових облікових записів.

2.6.1. Використані технології

Система автоматизації та управління транзакціями є повноцінним веб-додатком (Full-stack Web Application), розробленим із застосуванням комбінації сучасних технологій.

1. Бекенд (Backend).

Для реалізації серверної частини додатка використовується фреймворк Java Spring Boot. Spring Boot є потужним та широко застосовуваним фреймворком для розробки корпоративних Java-додатків. Він надає розширені можливості, включаючи вбудовану підтримку для створення RESTful веб-сервісів та інтеграції з базами даних, що дозволяє швидко розробляти масштабовані, надійні та безпечні програмні рішення. Використання Spring Boot сприяло ефективній реалізації складної бізнес-логіки, необхідної для управління транзакціями.

2. Фронтенд (Frontend).

Користувацький інтерфейс системи розроблено з використанням бібліотеки ReactJS. ReactJS є популярним JavaScript-фреймворком для

побудови інтерфейсів користувача, відомим своєю модульністю, гнучкістю та ефективністю. Застосування ReactJS дозволило створити односторінковий додаток (Single Page Application - SPA), що забезпечує плавний та безперебійний користувацький досвід, оскільки перехід між розділами додатка не вимагає повного перезавантаження сторінки.

3. База даних.

Для управління та зберігання даних про транзакції використовується реляційна система управління базами даних MySQL. MySQL є однією з найбільш розповсюджених СУБД для веб-додатків, яка відзначається високою масштабованістю, адаптивністю та потужними функціями управління даними, включаючи підтримку складних запитів та механізмів транзакцій.

4. Розгортання (Deployment).

Веб-додаток, що включає фронтенд, бекенд та базу даних, розміщений на платформі Google Cloud. Це забезпечує доступ до системи через публічний URL, що підвищує доступність та гнучкість розгортання.

2.6.2. Обґрунтування архітектурного вибору

Архітектурний стиль Model-View-Controller (MVC) був обраний для проєкту системи управління транзакціями у видавничій сфері завдяки своїй здатності забезпечувати чітке розділення відповідальності між компонентами системи. Таке розділення вирішує проблему відокремлення логіки відображення від логіки додатка та управління даними. У цій архітектурі, Модель представляє дані та бізнес-логіку системи, Вигляд відповідає за представлення даних та взаємодію з користувацьким інтерфейсом, а Контролер діє як посередник, обробляючи введення користувача, викликаючи відповідні дії моделі та оновлюючи вигляд за необхідності.

Даний підхід сприяє створенню високомодульного та легко керованого коду, що є критично важливим для складних проєктів, таких як система

управління транзакціями. Він підвищує підтримуваність системи, спрощує внесення змін та розширення функціоналу, а також забезпечує ефективну співпрацю між командами розробників, що працюють над різними компонентами.

2.7. Проектування діаграми класів

На рисунку 2.4 показано діаграму класів (Class Diagram), яка візуалізує структуру системи управління транзакціями у видавничій діяльності, демонструючи класи, їх атрибути, методи та взаємозв'язки.

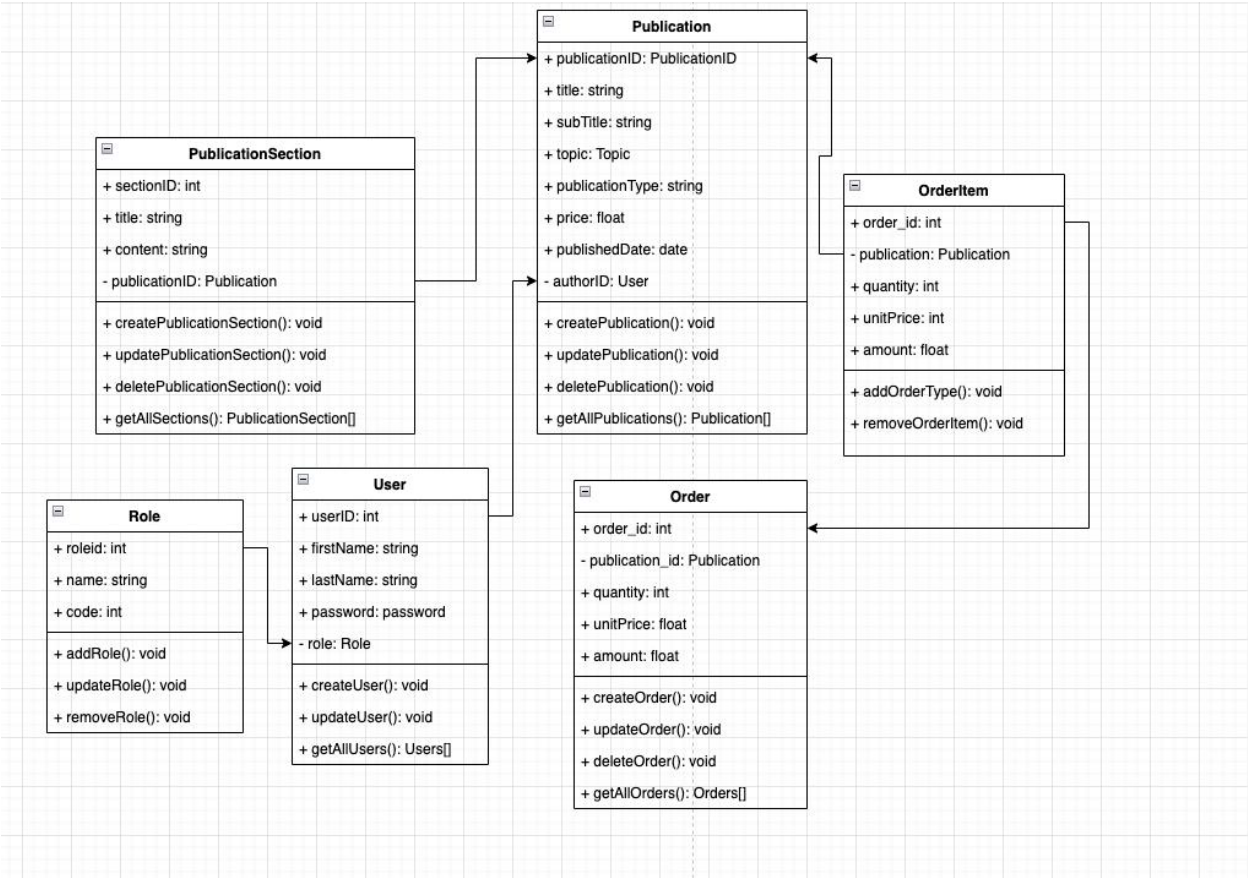

Рисунок 2.4 – Діаграма класів

Опис класів та їх атрибутів/методів:

1. Publication (Публікація):

- Атрибути:

- `publicationID`: Унікальний ідентифікатор публікації (тип `PublicationID`).

- `title`: Назва публікації (тип `string`).

- `subTitle`: Підзаголовок публікації (тип `string`).

- `topic`: Тематика публікації (тип `Topic` – інший клас або перелік).

- `publicationType`: Тип публікації (тип `string` – посилається на `PublicationType`).

- `price`: Ціна публікації (тип `float`).

- `publishedDate`: Дата публікації (тип `date`).

- `authorID`: Ідентифікатор автора (тип `User` – зв'язок з класом `User`).

- Методи:

- `createPublication()`: Створення нової публікації.

- `updatePublication()`: Оновлення існуючої публікації.

- `deletePublication()`: Видалення публікації.

- `getAllPublications()`: Отримання списку всіх публікацій.

2. PublicationSection (Розділ публікації):

- Атрибути:

- `sectionID`: Унікальний ідентифікатор розділу (тип `int`).

- `title`: Назва розділу (тип `string`).

- `content`: Вміст розділу (тип `string`).

- `publicationID`: Ідентифікатор публікації, до якої належить розділ (тип `Publication` – посилання на клас `Publication`).

- Методи:

- `createPublicationSection()`: Створення нового розділу публікації.

- `updatePublicationSection()`: Оновлення існуючого розділу публікації.

- `deletePublicationSection()`: Видалення розділу публікації.

- `getAllSections()`: Отримання списку всіх розділів (повертає `PublicationSection[]`).

3. User (Користувач):

- Атрибути:

- `userID`: Унікальний ідентифікатор користувача (тип `int`).

- `firstName`: Ім'я користувача (тип `string`).

- `lastName`: Прізвище користувача (тип `string`).

- `password`: Пароль користувача (тип `password` – хешований рядок).

- `role`: Роль користувача (тип `Role` – посилання на клас `Role`).

- Методи:

- `createUser()`: Створення нового користувача.

- `updateUser()`: Оновлення інформації про користувача.

- `getAllUsers()`: Отримання списку всіх користувачів (повертає `Users[]`).

4. Role (Роль):

- Атрибути:

- `roleID`: Унікальний ідентифікатор ролі (тип `int`).

- `name`: Назва ролі (тип `string`).

- `code`: Код ролі (тип `int`).

- Методи:

- `addRole()`: Додавання нової ролі.

- `updateRole()`: Оновлення існуючої ролі.

- `removeRole()`: Видалення ролі.

5. Order (Замовлення):

- Атрибути:

- `order_id`: Унікальний ідентифікатор замовлення (тип `int`).

- `publication_id`: Ідентифікатор публікації, що входить до замовлення (тип `Publication` – посилання на клас `Publication`).

- `quantity`: Кількість замовлених одиниць (тип `int`).

- `unitPrice`: Ціна за одиницю публікації (тип `float`).

- `amount`: Загальна сума замовлення (тип `float`).

- Методи:

- `createOrder()`: Створення нового замовлення.

- `updateOrder()`: Оновлення існуючого замовлення.

- `deleteOrder()`: Видалення замовлення.

- `getAllOrders()`: Отримання списку всіх замовлень (повертає `Orders[]`).

6. OrderItem (Елемент замовлення):

- Атрибути:

- `order_id`: Ідентифікатор замовлення, до якого належить елемент (тип `int`).

- `publication`: Об'єкт публікації, що входить до елемента замовлення (тип `Publication` – посилання на клас `Publication`).

- `quantity`: Кількість одиниць даної публікації в елементі замовлення (тип `int`).

- `unitPrice`: Ціна за одиницю публікації в елементі замовлення (тип `int` – ймовірно, має бути `float` або `decimal`).

- `amount`: Загальна вартість даного елемента замовлення (тип `float`).

- Методи:

- `addOrderType()`: Додавання типу замовлення (додавання нового елемента до замовлення або типу елемента).

- `removeOrderItem()`: Видалення елемента замовлення.

Взаємозв'язки між класами:

- Publication - PublicationSection:

- Одностороння асоціація: `PublicationSection` має посилання на `Publication`. Це означає, що розділ належить до певної публікації. Це, ймовірно, зв'язок "один-до-багатьох" (одна публікація може мати багато розділів), хоча явно не вказано множинність на кінці `Publication`.

- User - Role:

- Агрегація (або композиція): `User` має атрибут `role` типу `Role`. Це зв'язок "один-до-одного" або "один-до-багатьох" (одна роль може бути призначена багатьом користувачам, але один користувач має одну роль). Діаграма показує `role: Role` всередині `User`, що зазвичай інтерпретується як композиція або агрегація.

- Order - Publication:

- Асоціація: `Order` має атрибут `publication_id` типу `Publication`. Це вказує на те, що замовлення стосується певної публікації. Множинність не вказана явно, але це "багато-до-одного" (багато замовлень на одну публікацію).

- Order - OrderItem:

- Композиція/Агрегація: `Order` має посилання на `OrderItem` (через атрибут `order_id` в `OrderItem`, хоча зв'язок не зображено явно як лінія між класами, а через атрибут). Це означає, що замовлення складається з одного або декількох елементів замовлення. Це зв'язок "один-до-багатьох" (одне замовлення має багато елементів).

- OrderItem - Publication:

- Асоціація: `OrderItem` має атрибут `publication` типу `Publication`. Це вказує, що кожен елемент замовлення стосується певної публікації.

Діаграма класів надає уявлення про статичну структуру системи, визначаючи сутності, їхні характеристики та операції, які вони можуть виконувати. Вона показує логічну модель даних, яка буде реалізована в базі даних, а також об'єкти бізнес-логіки.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ТА УПРАВЛІННЯ ТРАНЗАКЦІЯМИ У ВИДАВНИЧІЙ ДІЯЛЬНОСТІ

3.1. Розробка модулів аутентифікації та панелі управління

Цей розділ присвячений детальному опису ключових функціональних модулів Системи автоматизації та управління транзакціями, а саме: сторінки входу (автентифікації) та панелі управління.

3.1.1. Сторінка входу – механізм аутентифікації

Сторінка входу функціонує як основна точка доступу до системи управління транзакціями, вимагаючи від користувачів введення автентифікаційних даних для отримання дозволу на вхід. Після того, як користувач вводить свій логін та пароль, на сервер надсилається запит на аутентифікацію. У випадку, якщо надані облікові дані виявляються недійсними, системі ініціюється виведення відповідного сповіщення про некоректність введеної інформації.

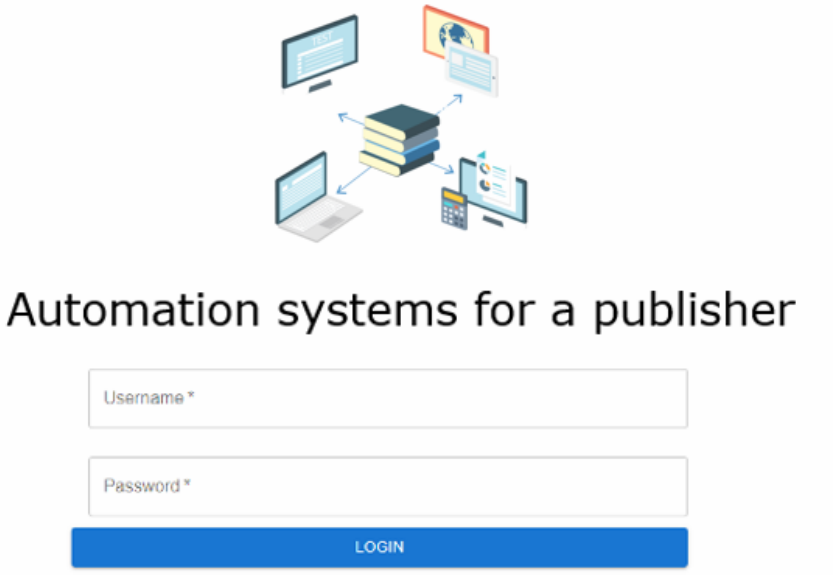

Рисунок 3.1 – Сторінка входу в систему

У разі успішної перевірки облікових даних сервер генерує токен автентифікації, що містить інформацію про користувача, включаючи його роль. Одночасно створюється сесія користувача на бекенді. Згенерований токен повертається клієнтській стороні та в подальшому використовується для всіх авторизованих запитів до сервера. Успішна автентифікація призводить до автоматичного перенаправлення користувача на інтерфейс панелі управління, яка надає доступ до всіх функцій та можливостей системи відповідно до його ролі. Сторінка входу відіграє ключову роль у забезпеченні безпеки системи та оптимізації взаємодії з користувачами.

3.1.2. Панель управління

Панель управління є центральним користувацьким інтерфейсом Системи автоматизації та управління транзакціями, що надає користувачам доступ до всіх її модулів. Після успішної автентифікації користувач автоматично перенаправляється на панель управління, де функціональні можливості адаптуються відповідно до його призначеної ролі. Дизайн панелі управління орієнтований на інтуїтивність та зручність використання, характеризуючись чітким та логічно організованим макетом.

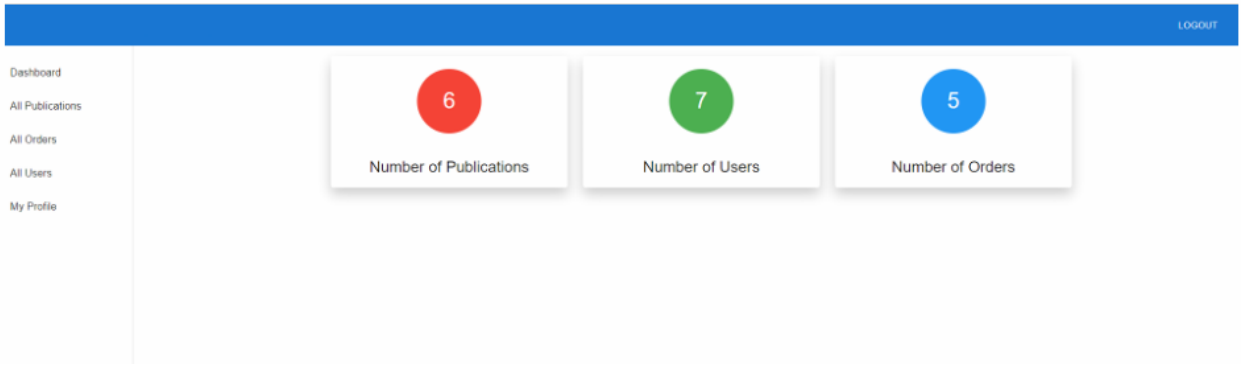

Рисунок 3.2 – Вигляд панелі управління (дашборд)

Однією з ключових особливостей панелі управління є реалізація контролю доступу на основі ролей (Role-Based Access Control - RBAC). Цей

механізм гарантує, що кожен користувач має доступ виключно до тих модулів, які відповідають його функціональним обов'язкам. Наприклад:

- Автори мають можливість переглядати свої публікації в модулі "Всі публікації".

- Редактори в цьому ж модулі можуть управляти та модифікувати публікації.

- Адміністратори отримують доступ до модуля "Всі замовлення" для управління та відстеження всіх транзакцій.

- Дистриб'ютори використовують модуль "Всі замовлення" для розміщення нових замовлень.

Додатково, через панель управління користувачам доступні аналітичні інструменти, які сприяють глибшому розумінню операційної діяльності та прийняттю рішень на основі даних. Спеціалізована сторінка аналітики надає агреговані дані про використання системи, включаючи:

- Загальну кількість публікацій, що знаходяться в системі.

- Кількість зареєстрованих користувачів.

- Обсяг замовлень, розміщених дистриб'юторами.

Інформація для цих метрик отримується шляхом виконання відповідних запитів до бази даних MySQL, забезпечуючи актуальність та достовірність представлених аналітичних даних.

3.2. Модулі управління контентом у системі автоматизації видавничої діяльності

У рамках системи автоматизації видавничої діяльності (САВД) розроблені спеціалізовані модулі для ефективного управління контентом, що охоплюють створення, редагування та каталогізацію публікацій та їхніх складових.

3.2.1. Модуль "Публікації"

Модуль "Публікації" надає авторам комплексну платформу для створення та управління їхніми творами. Основний функціонал модуля дозволяє авторам ініціювати нові публікації, вказуючи їхню назву та тип (наприклад, книга, журнал). Після створення публікації автор може послідовно додавати до неї окремі розділи. Кожен розділ характеризується унікальною назвою та описом, які можуть бути відредаговані автором у будь-який час.

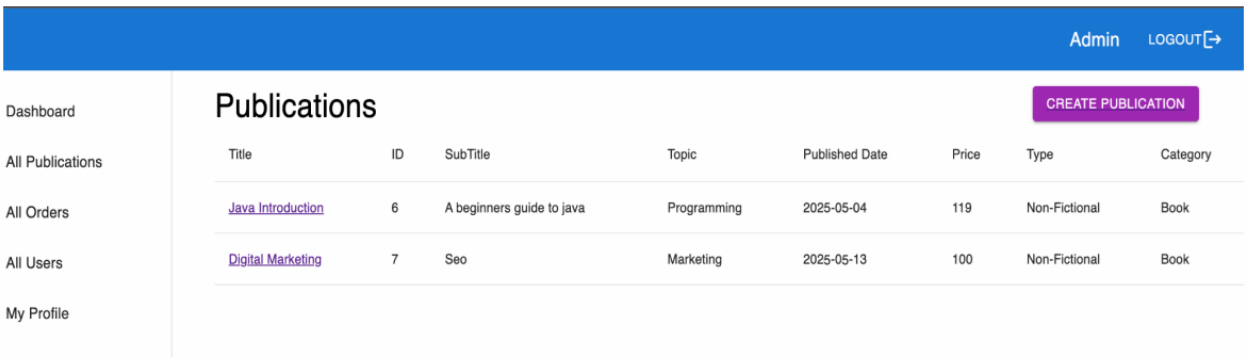

Рисунок 3.3 – Перелік публікацій

При створенні нової публікації, автор також зобов'язаний надати метадані, що включають жанр, категорію та бажану ціну для видання. Додатково, система передбачає можливість асоціювання публікації з певними ключовими словами для покращення індексації та пошуку.

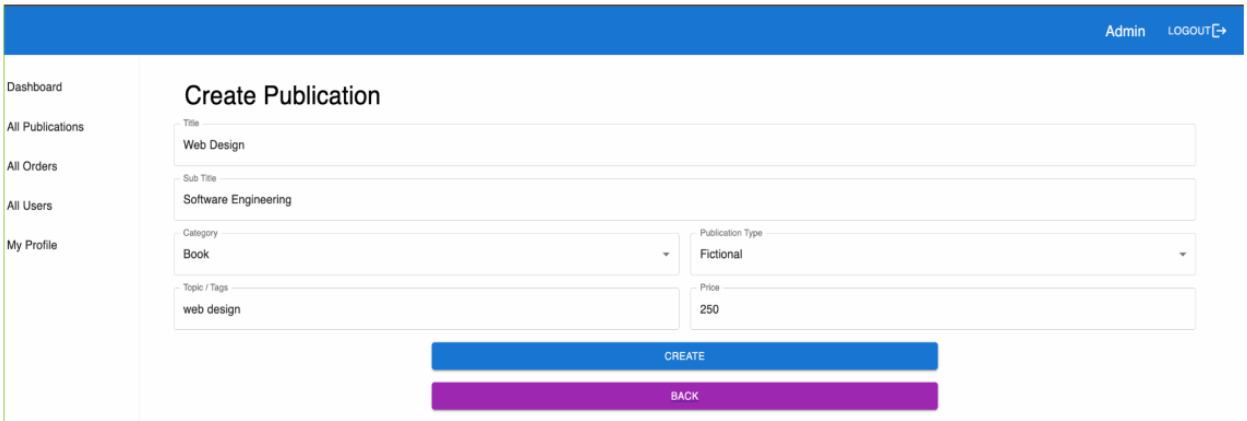

Рисунок 3.4 – Форма створення публікації

Центральна частина екрану присвячена формі для створення нової публікації. Форма містить такі поля для введення інформації:

- Title (Назва).
- Sub Title (Підзаголовок).
- Category (Категорія): Поле вибору (випадаючий список).
- Publication Type (Тип публікації): Поле вибору (випадаючий список).
- Topics / Tags (Теми / Теги).
- Price (Ціна).

Під полями форми розташовані дві кнопки:

- "CREATE" (Створити) – синього кольору для відправки форми.
- "BACK" (Назад) – фіолетового кольору для повернення на попередню сторінку.

Для забезпечення цілісності та коректності даних, у модулі "Публікації" впроваджено механізми валідації форм на декількох етапах. Наприклад, при створенні нової публікації система вимагає обов'язкового вибору типу публікації та введення назви. Відправка форми блокується до заповнення всіх обов'язкових полів.

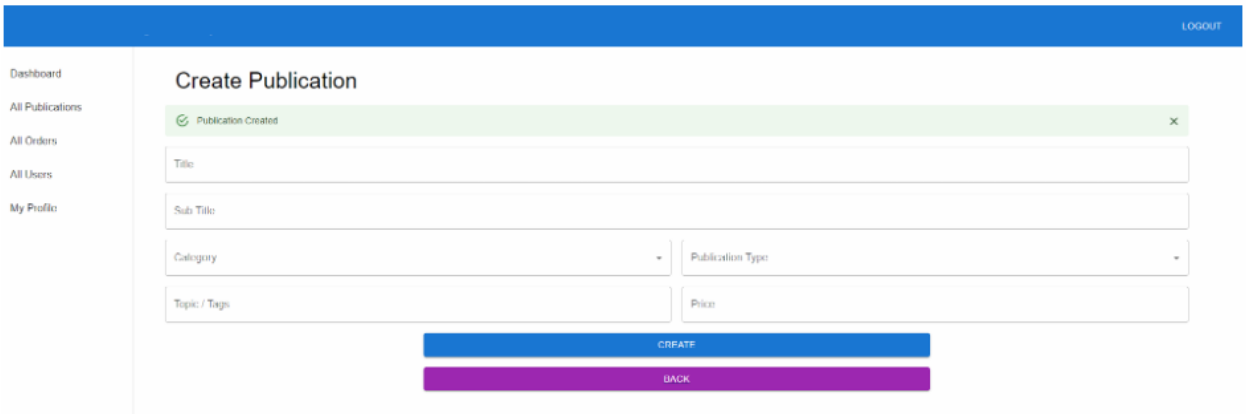

Рисунок 3.5 – Повідомлення про створену публікацію

Важливою функціональною можливістю є підтримка динамічної зміни розділів публікації. Автор зберігає можливість модифікувати вміст будь-якого розділу навіть після офіційної публікації матеріалу. Це забезпечує

актуальність контенту та дозволяє авторам постійно вдосконалювати свої роботи.

3.2.2. Модуль "Розділи публікацій"

Модуль "Розділи публікацій" є інтегральною частиною системи, що дозволяє авторам створювати, оновлювати та ефективно управляти окремими розділами в межах публікації. Кожен розділ може мати індивідуальні атрибути, такі як тип розділу, назва розділу та вміст розділу.

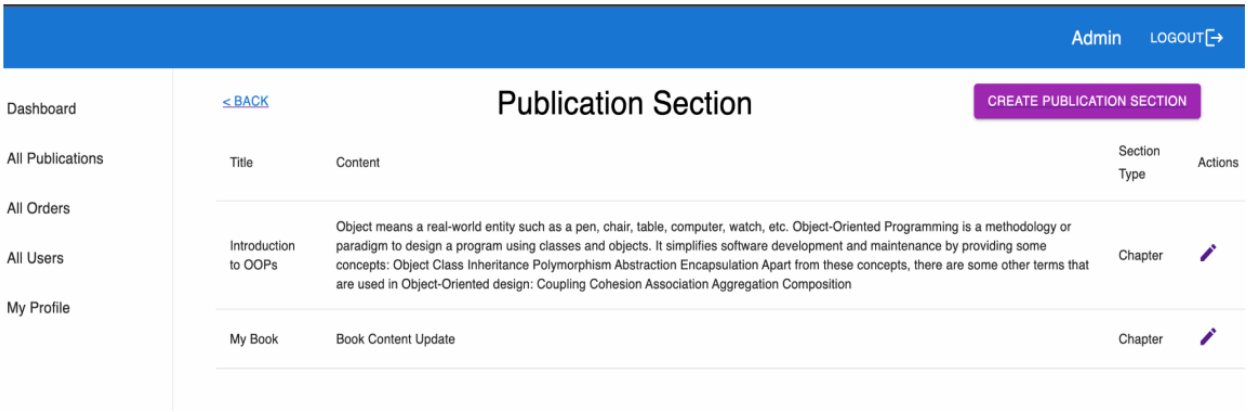

Рисунок 3.6 – Розділи публікацій

Процес створення розділів інтегрований у функціонал додавання нової публікації, де автор вказує тип, назву та зміст кожного нового розділу. Автор має повну гнучкість у редагуванні цих розділів, включаючи зміну назви або основного тексту. Крім того, функціонал управління розділами поширюється і на редакторів публікацій. Редактори мають право вносити зміни до існуючих розділів, додавати нові або видаляти наявні, забезпечуючи повноцінний контроль над структурою та змістом публікації.

Модуль "Розділи публікацій" також містить ряд валідаційних перевірок форм для гарантування точності та відповідності наданої інформації. Це включає перевірку на відсутність порожнього поля для назви розділу та обмеження на максимальну довжину вмісту розділу.

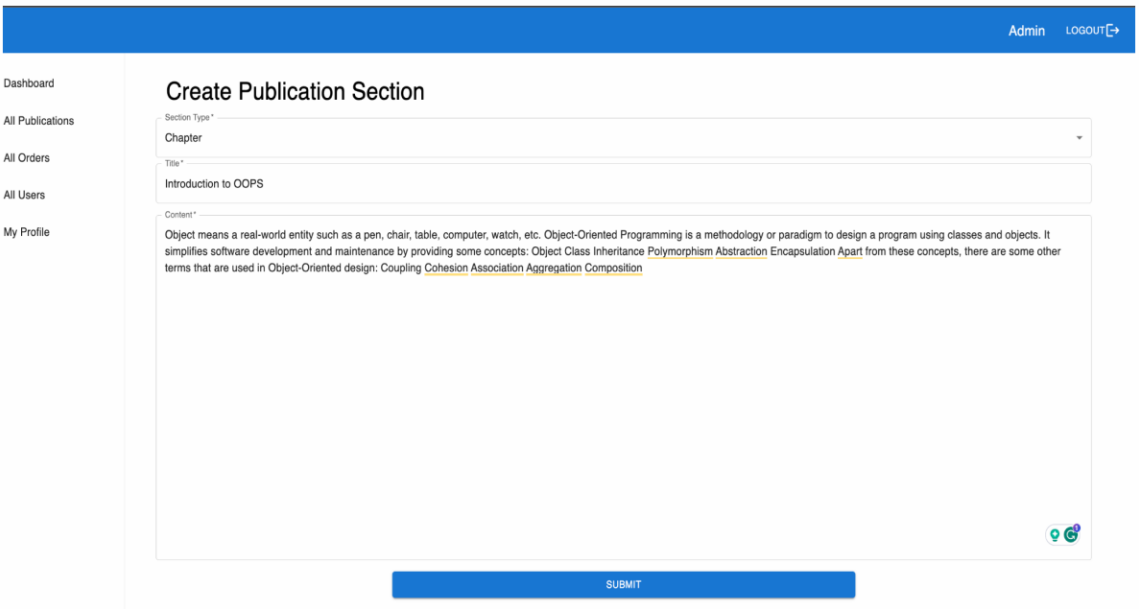

Рисунок 3.7 - Форма розділу "Створення публікації"

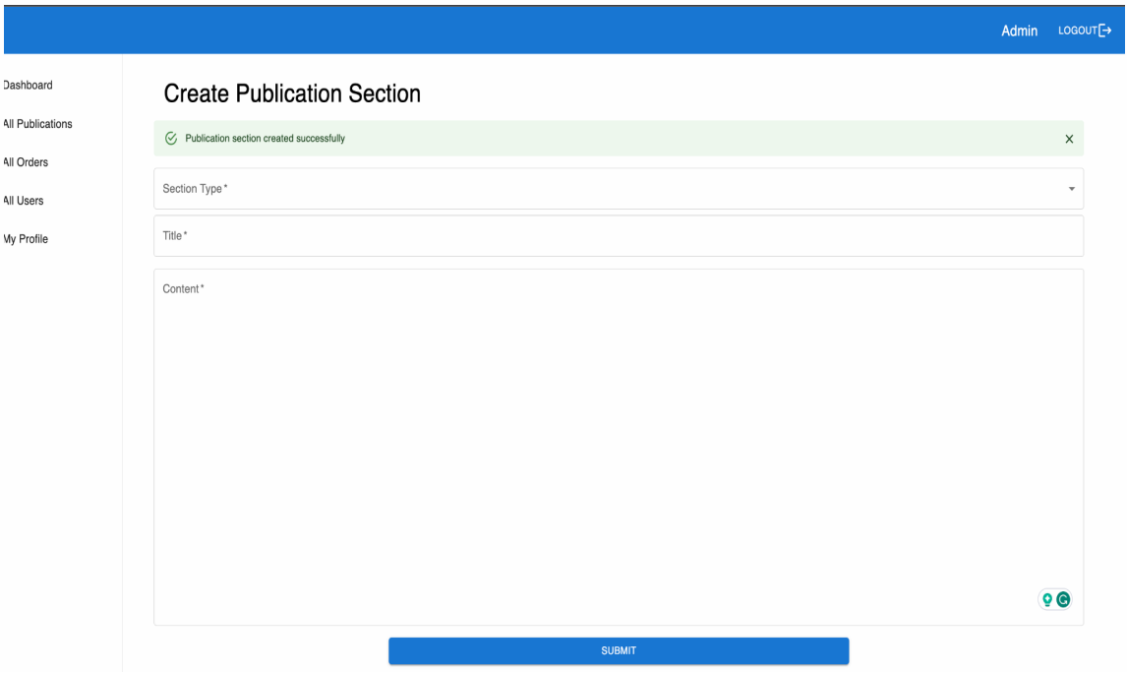

Рисунок 3.8 - Створено розділ "Публікація"

3.3. Функціонал управління замовленнями у системі автоматизації видавничої діяльності

У рамках системи автоматизації видавничої діяльності реалізовано модуль управління продажами, який надає дистриб'юторам розширені

можливості для ефективного формування та обробки замовлень на друковану продукцію.

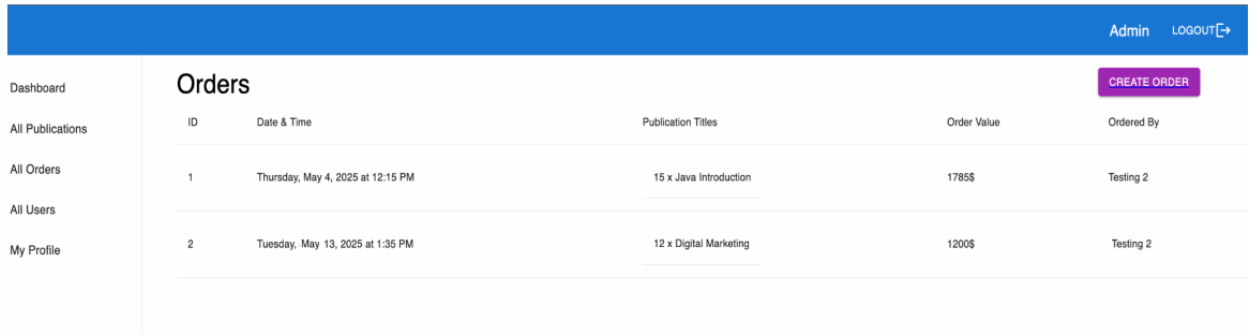

Рисунок 3.9 – Перелік замовлень

Для ініціації нового замовлення дистриб'ютору необхідно надати базовий набір інформації, що включає:

Ідентифікатор публікації (Publication ID): Дистриб'ютор повинен ввести унікальний ідентифікатор бажаної публікації. Цей ідентифікатор слугує первинним ключем для доступу до відповідних даних у таблиці публікацій бази даних. Такий підхід забезпечує точну ідентифікацію та вибір коректного видання для замовлення.

Замовлена кількість (Quantity): Дистриб'ютор вказує необхідну кількість копій обраної публікації.

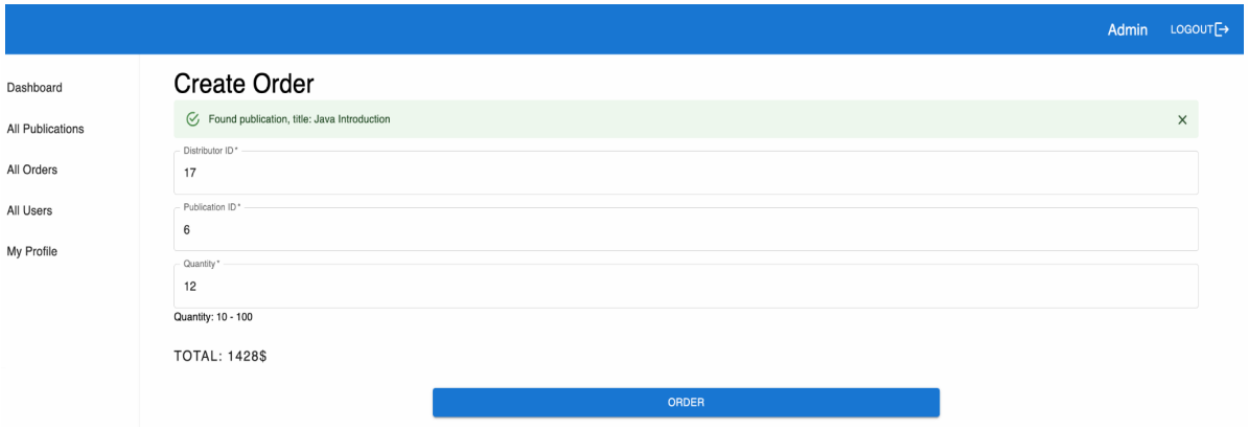

Рисунок 3.10 - Перевірка ідентифікатора публікації

На основі наданої кількості та цінової інформації, отриманої з таблиці публікацій (де зберігається ціна за одиницю), система автоматично розраховує загальну вартість замовлення. Цей автоматизований розрахунок підвищує ефективність та мінімізує ймовірність помилок у фінансових операціях.

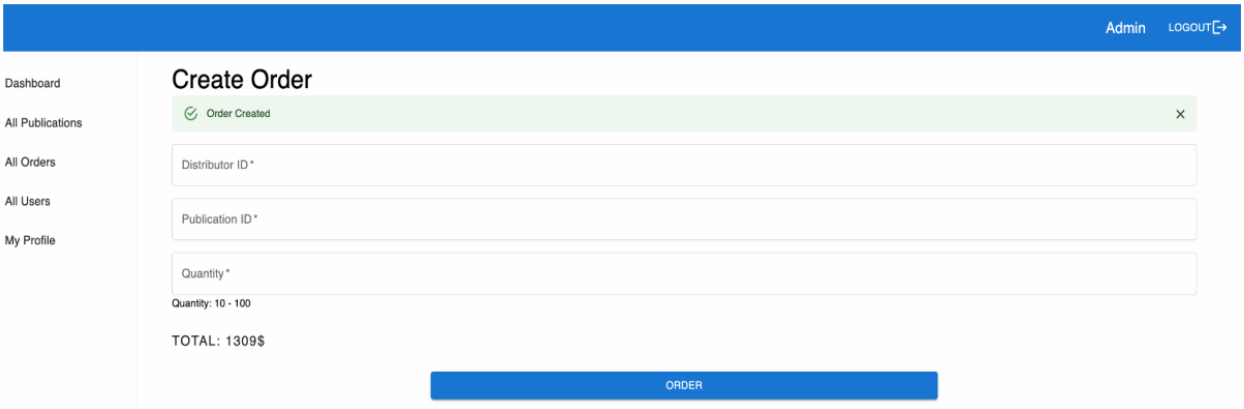

Рисунок 3.11 - Замовлення успішно створено

Функціонал модуля управління замовленнями дозволяє дистриб'юторам не тільки розміщувати нові замовлення, але й ефективно управляти вже існуючими замовленнями. Це включає можливість перегляду статусу замовлень, відстеження їх виконання та, за необхідності, внесення коригувань, якщо це передбачено бізнес-процесами. Таким чином, система забезпечує повний життєвий цикл управління замовленнями для дистриб'юторів.

3.4. Модуль управління користувачами

У рамках системи автоматизації видавничої діяльності передбачені спеціалізовані модулі для управління користувачами, що забезпечують контроль доступу та персоналізацію інформації.

3.4.1. Модуль "Користувачі"

Модуль "Користувачі" є критично важливим компонентом системи, доступ до якого обмежений виключно адміністратором. Цей модуль надає повний перелік усіх зареєстрованих користувачів додатку, включаючи детальну інформацію про кожного з них. При реєстрації нового користувача адміністратор зобов'язаний ввести наступні дані: ім'я користувача (логін), ім'я, прізвище, пароль та призначити роль користувача.

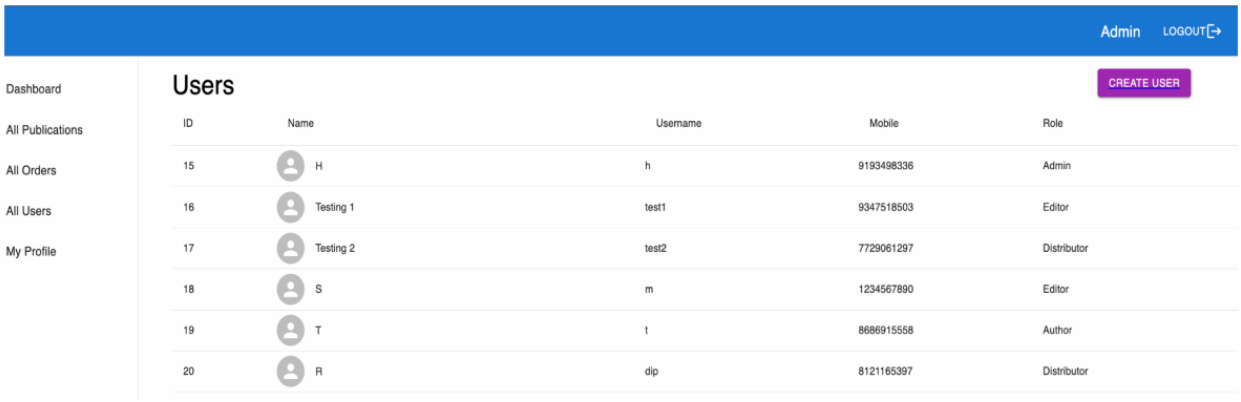

Рисунок 3.12 – Перелік користувачів

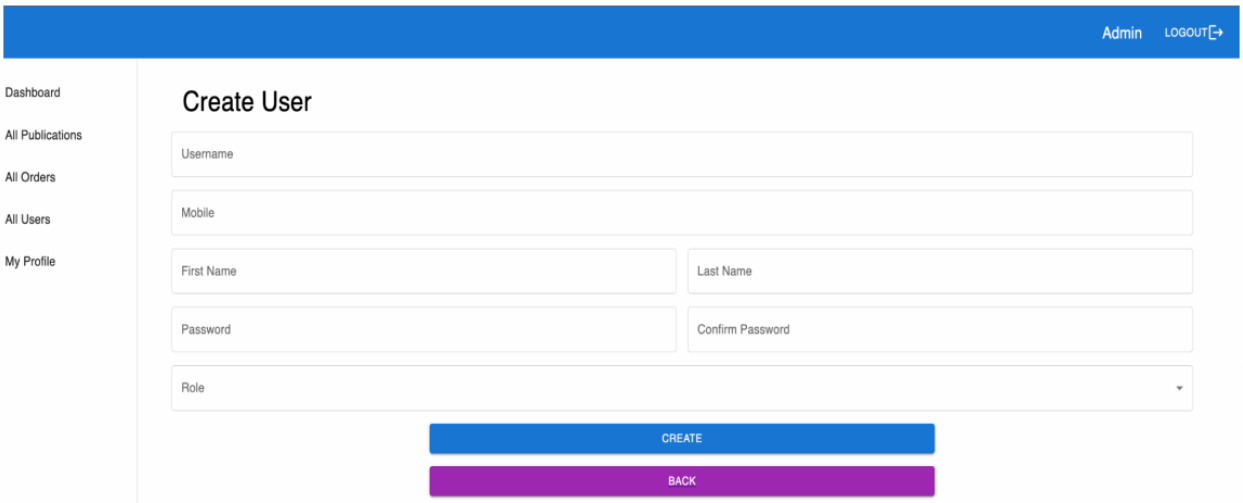

Рисунок 3.13 – Форма створення користувача

Модуль "Користувачі" також надає адміністратору функціонал для модифікації або видалення інформації про користувачів. Наприклад, адміністратор може деактивувати обліковий запис користувача, який більше

не є активним учасником системи, або змінити його роль, наприклад, з "письменника" на "автора". Ця функція забезпечує повний контроль над доступом до системи, гарантуючи, що лише авторизовані користувачі мають відповідні дозволи.

3.4.2. Модуль "Мій профіль"

Модуль "Мій профіль" призначений для надання користувачам можливості перегляду та модифікації власної особистої інформації.

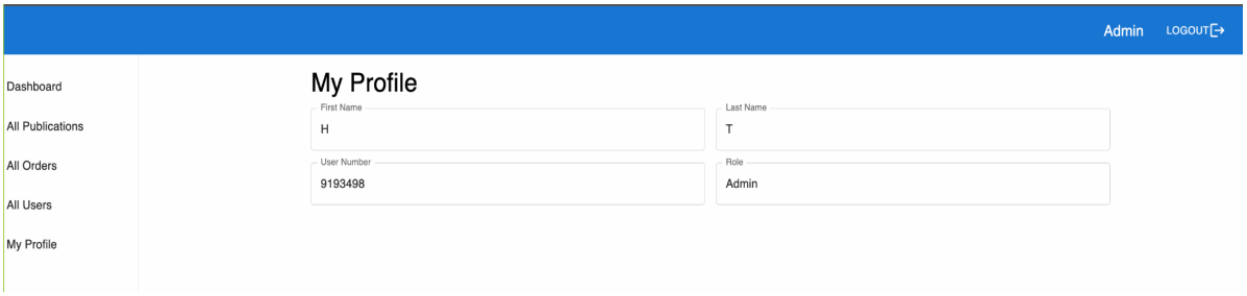

Рисунок 3.14 – Форма оновлення профілю користувача

Цей функціонал є доступним для всіх зареєстрованих користувачів системи. Користувачі можуть отримати доступ до цього розділу через відповідну кнопку або посилання, позначене як "Мій профіль" в інтерфейсі системи.

3.5. Деталізація архітектури та технологій бекенду розроблюваної системи

Цей розділ присвячений детальному опису архітектури бекенду системи, що реалізує клієнт-серверну модель, а також принципи її функціонування та ключові компоненти.

3.5.1. Архітектура бекенду

Архітектура системи базується на клієнт-серверній моделі. Клієнтська частина системи реалізована за допомогою фреймворку ReactJS, тоді як серверна частина розроблена на основі Java Spring Boot. Як система управління базами даних (СУБД) використовується MySQL.

Взаємодія між клієнтською та серверною частинами програми здійснюється за допомогою RESTful API-викликів. Сервер відповідає за обробку цих API-запитів, а також за операції з даними в базі даних: отримання, модифікацію та зберігання.

Для забезпечення безпечної комунікації між клієнтом і сервером впроваджено механізм авторизації на основі JSON Web Tokens (JWT), реалізований за допомогою Spring Security. Це гарантує, що доступ до функціоналу програми мають лише авторизовані користувачі. Архітектура системи спроєктована з урахуванням найкращих галузевих практик, з особливим акцентом на гнучкість, легкість управління та масштабованість. Розглянемо компоненти бекенду

3.5.2. Сутності (Entities)

Сутності є Java-об'єктами, що представляють структури даних, які зберігатимуться в базі даних. Кожна сутність, як правило, відображається на відповідну таблицю в реляційній базі даних, а екземпляри сутності – на рядки таблиці. У системі СУТ визначено наступні сутності:

- Order (Замовлення)
- OrderItem (Елемент замовлення)
- Publication (Публікація)
- PublicationCategory (Категорія публікації)
- PublicationSection (Розділ публікації)
- PublicationSectionType (Тип розділу публікації)
- PublicationType (Тип публікації)

- User (Користувач)
- UserRole (Роль користувача)

Наведений нижче фрагмент коду ілюструє структуру сутності Order.

```java
@Data // Автоматично генерує геттери, сеттери, toString, equals та hashCode
@Entity // Позначає клас як сутність JPA
@Table(name = "orders") // Визначає назву таблиці в базі даних
public class Order {

    @Id // Позначає поле як первинний ключ
    @GeneratedValue(strategy = GenerationType.IDENTITY) // Автоматичне генерування зн
    private Long id;

    @ManyToOne // Зв'язок "багато до одного" з сутністю User
    @JoinColumn(name = "receiver_id", nullable = false) // Вказує на стовпець зовніш
    private User receiver; // Користувач, який розмістив замовлення

    @Column(nullable = false) // Поле не може бути null
    private Double amount = 0.0; // Загальна сума замовлення

    @Enumerated(EnumType.STRING) // Зберігає перелічення як рядок у базі даних
    private OrderStatus status = OrderStatus.PROCESSING; // Поточний статус замовлен

    @OneToMany(mappedBy = "order", cascade = CascadeType.ALL) // Зв'язок "один до ба
    private List<OrderItem> items = new ArrayList<>(); // Список елементів у замовле

    @CreationTimestamp // Автоматично встановлює час створення запису
    private Date createdAt;

    @UpdateTimestamp // Автоматично оновлює час при зміні запису
    private Date updatedAt;
}
```

Рисунок 3.15 - Структура сутності Order

3.5.3 Контролери (Controllers)

Контролери відповідають за обробку вхідних HTTP-запитів та формування відповідних HTTP-відповідей для клієнта. Вони визначають методи, що відповідають певним HTTP-методам (GET, POST, PUT, DELETE) та URL-шляхам. Контролери в Spring Boot зазвичай анотуються @Controller або @RestController. У системі розроблено наступні контролери:

- OrderController
- PublicationCategoryController
- PublicationController

- PublicationSectionController
- PublicationSectionTypeController
- PublicationTypeController
- UserController
- UserRoleController
- AuthenticationController (для обробки API входу та генерації JWT)

Наведений нижче фрагмент коду демонструє структуру OrderController.

```
@RestController // Позначає клас як REST контролер
@RequestMapping("/orders") // Базовий URL для всіх методів контролера
@Slf4j // Автоматично генерує логер
public class OrderController {

    @Autowired // Автоматичне впровадження залежностей
    private OrderService orderService; // Сервіс для обробки бізнес-логіки замовлень

    @GetMapping() // Обробляє GET-запити до /orders
    public ResponseEntity<List<OrderDTO>> fetch() {
        log.info(msg("In get orders")); // Логування початку операції
        List<OrderDTO> orders = orderService.fetch(); // Виклик сервісу для отриманн
        log.info(format("Get order", orders)); // Логування результату
        return new ResponseEntity<List<OrderDTO>>(orders, HttpStatus.OK); // Поверн
    }

    @GetMapping("/{id}") // Обробляє GET-запити до /orders/{id}
    public ResponseEntity<OrderDTO> getById(@PathVariable Long id) { // Отримання I
        OrderDTO orderDTO = orderService.findById(id); // Виклик сервісу для пошуку
        return new ResponseEntity<OrderDTO>(orderDTO, HttpStatus.OK); // Повернення
    }

    @PostMapping() // Обробляє POST-запити до /orders
    public ResponseEntity<SuccessResponse> create(@Valid @RequestBody OrderRequest
        orderService.create(orderRequest); // Виклик сервісу для створення замовлення
        SuccessResponse response = new SuccessResponse(); // Створення об'єкта відпо
        response.setMessage(Constants.ORDER_CREATED_SUCCESSFULLY); // Встановлення
        return new ResponseEntity<SuccessResponse>(response, HttpStatus.CREATED);
    }
}
```

Рисунок 3.16 - Структура OrderController

3.5.4. Сервіси (Services)

Сервіси є класами, які інкапсулюють основну бізнес-логіку програми. Вони надають рівень абстракції між рівнем доступу до даних (що взаємодіє безпосередньо з базою даних або іншими зовнішніми системами) та

контролерами, які обробляють вхідні запити. У СУТ присутні наступні сервіси:

- OrderService
- PublicationCategoryService
- PublicationSectionService
- PublicationSectionTypeService
- PublicationService
- PublicationTypeService
- UserRoleService
- UserService

Наведений нижче фрагмент коду демонструє структуру OrderService:

```
@Service // Позначає клас як сервісний компонент Spring
@Slf4j // Автоматично генерує логер
public class OrderService {

    @Autowired
    private OrderRepository orderRepository; // Репозиторій для доступу до даних замо

    @Autowired
    private UserRepository userRepository; // Репозиторій для доступу до даних корист

    @Autowired
    private ModelMapper modelMapper; // Інструмент для мапінгу об'єктів

    public List<OrderDTO> fetch() {
        List<Order> orders = orderRepository.findAll(); // Отримання всіх замовлень
        return orders.stream().map(order -> modelMapper.map(order, OrderDTO.class)
    }

    public OrderDTO findById(Long id) {
        Order order = orderRepository.findById(id).orElseThrow(() -> new ResourceN
        return modelMapper.map(order, OrderDTO.class); // Перетворення сутності на D
    }

    public Order create(OrderRequest orderRequest) {
        User receiver = userRepository.findById(orderRequest.getReceiverId()).orEl
            new ResourceNotFoundException("User", "id", orderRequest.getReceiverId
        Order order = new Order(); // Створення нового об'єкта замовлення
        order.setReceiver(receiver); // Встановлення отримувача
        order.setAmount(orderRequest.getAmount()); // Встановлення суми
        order.setStatus(OrderStatus.PROCESSING); // Встановлення початкового статусу
        return orderRepository.save(order); // Збереження нового замовлення в базі да
    }
}
```

Рисунок 3.17 - Структура OrderService

3.6. Проектування діаграми послідовностей

Цей підрозділ представляє аналіз функціональних потоків системи за допомогою діаграм послідовностей. Ці діаграми ілюструють динамічну поведінку системи, відображаючи порядок взаємодії між різними акторами (користувачами) та компонентами системи під час виконання ключових операцій.

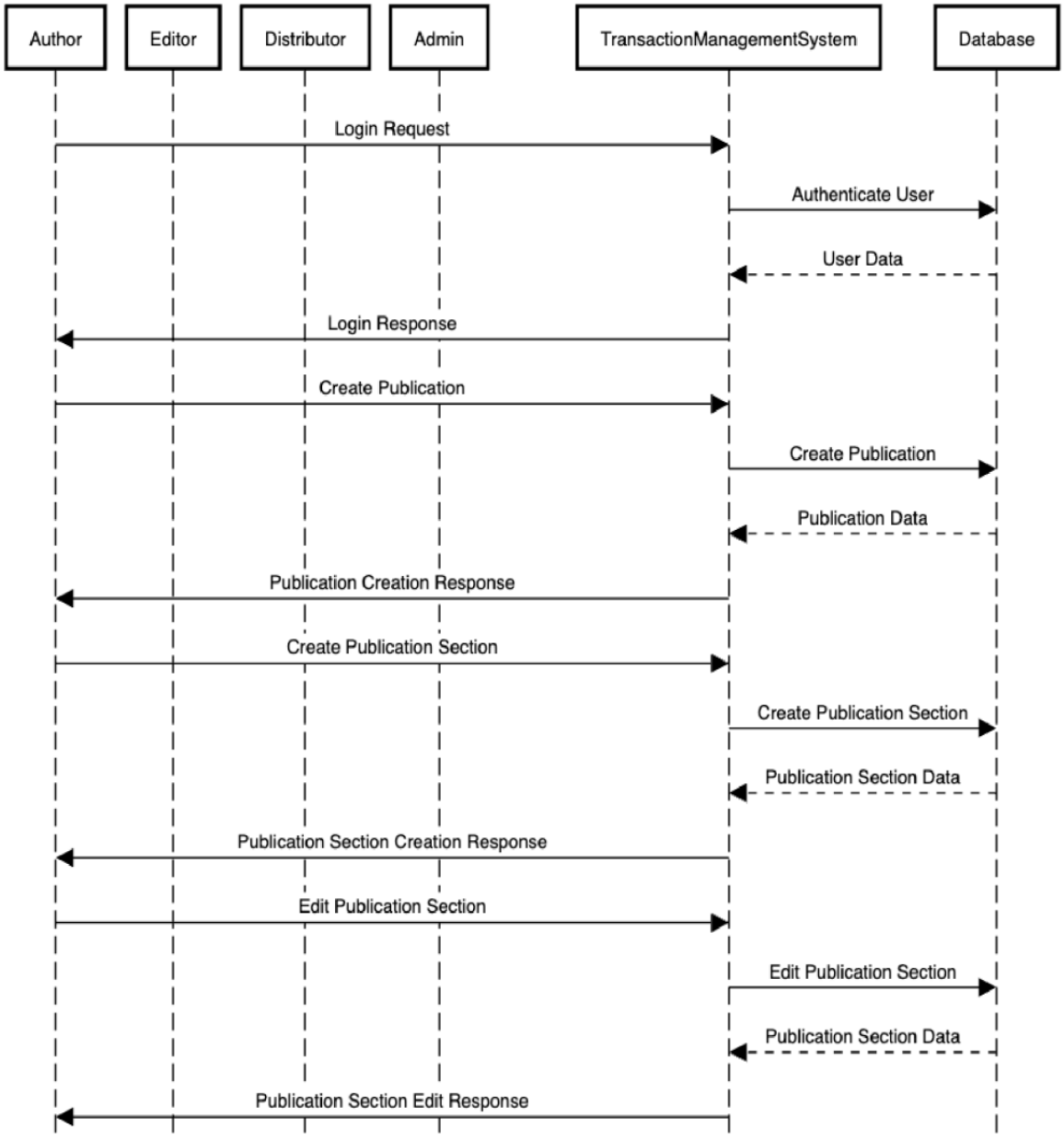

Рисунок 3.18 – Діаграма послідовності (початок)

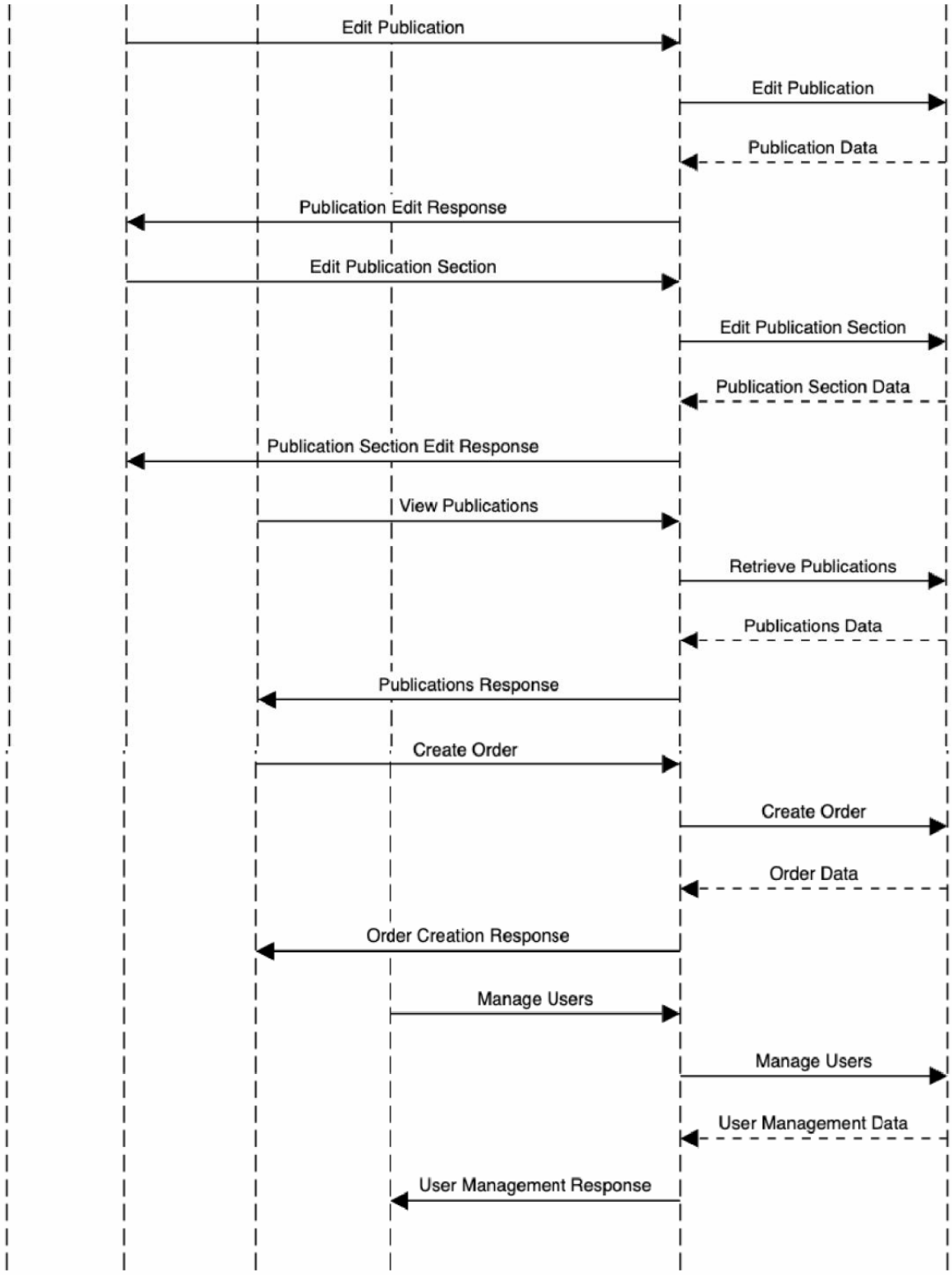

Рисунок 3.18 – Діаграма послідовності (закінчення)

Наведена серія діаграм послідовностей деталізує робочий процес створення публікацій та обробки замовлень, а також відображає взаємодії,

що відбуваються між різними користувацькими ролями та самою Системою управління транзакціями.

Ключові функціональні потоки та взаємодії акторів:

- Аутентифікація користувача: На початковому етапі, після входу користувача в систему, відбувається процес аутентифікації облікових даних. Цей крок є обов'язковим для всіх акторів і забезпечує контроль доступу до подальших функціональних можливостей.

- Управління публікаціями:

Автор: Після успішної автентифікації, автор має можливість створювати нові публікації та їхні складові розділи.

Редактор: Редактор, у свою чергу, уповноважений модифікувати існуючі публікації та їхні розділи, що дозволяє здійснювати контроль якості та вносити необхідні правки.

- Управління замовленнями:

Дистриб'ютор: Ця роль має доступ до перегляду всіх наявних публікацій у системі та можливість розміщувати замовлення на будь-яку з них, ініціюючи процес закупівлі.

Адміністратор: Адміністратор володіє розширеними правами, що дозволяють йому не тільки створювати замовлення, але й здійснювати управління користувачами та їхніми ролями в системі, забезпечуючи адміністративний контроль.

Діаграма послідовностей фокусується на демонстрації різноманітних дій, які користувачі можуть виконувати в системі, а також на послідовності взаємодій з внутрішніми компонентами системи, необхідних для виконання цих завдань.

РОЗДІЛ 4. ПЛАН ТЕСТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ У ВИДАВНИЧІЙ ДІЯЛЬНОСТІ

Цей розділ представляє план тестування системи автоматизації, сфокусований на системному рівні та базований на визначених функціональних та нефункціональних вимогах. Викладені тест-кейси є репрезентативними прикладами для верифікації ключових функціональних можливостей.

4.1. Тест-кейси системного рівня

Нижче наведено вибірку тест-кейсів системного рівня, розроблених на основі вимог та специфікацій СУТ:

4.1.1. Функціональність автентифікації користувача

Тест-кейс 1.1: Вхід з коректними обліковими даними.

Дії: Ввести валідні логін та пароль у відповідні поля форми входу.

Очікуваний результат: Користувач успішно автентифікований та перенаправлений на панель управління відповідно до своєї ролі.

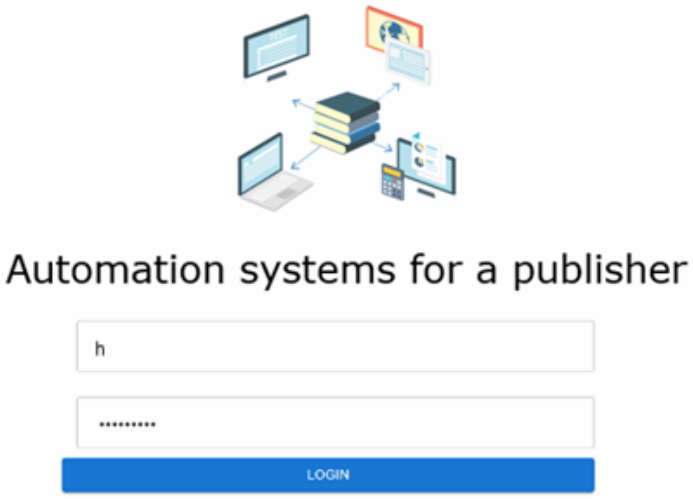

Рисунок 4.1 - Вхід з коректними обліковими даними

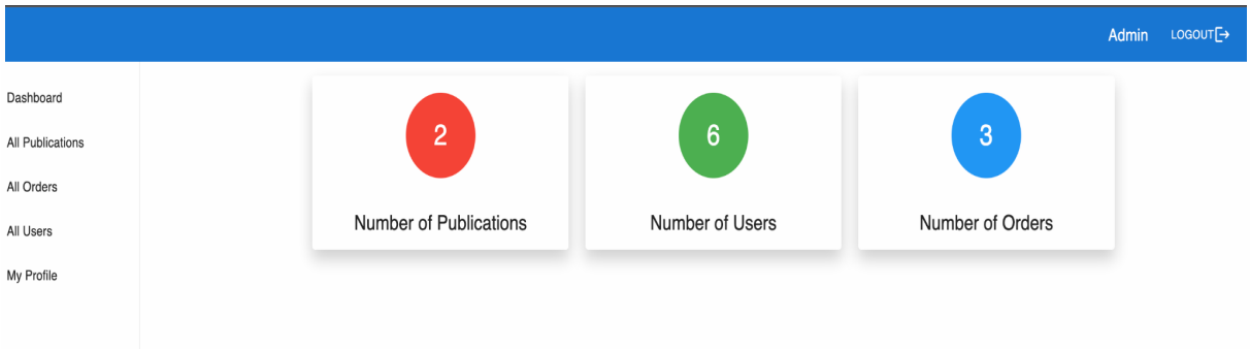

Рисунок 4.2 – Отримана функціональність після входу авторизованого користувача

Тест-кейс 1.2: Вхід з некоректними обліковими даними.

Дії: Ввести невалідні логін та/або пароль у відповідні поля форми входу.

Очікуваний результат: Система відмовляє в доступі та відображає повідомлення про помилку автентифікації (наприклад, "Неправильний логін або пароль").

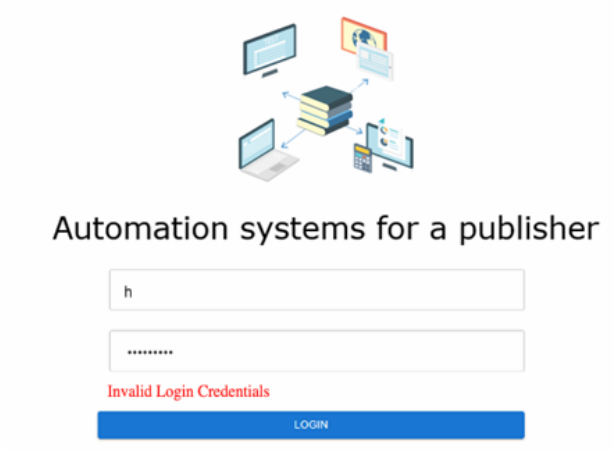

Рисунок 4.3 - Вхід з некоректними обліковими даними

4.1.2. Функціональність створення публікації

Тест-кейс 2.1: Створення публікації з усіма обов'язковими полями.

Дії: Заповнити всі обов'язкові поля форми створення публікації коректними даними та ініціювати збереження.

Очікуваний результат: Публікація успішно створена та додана до системи.

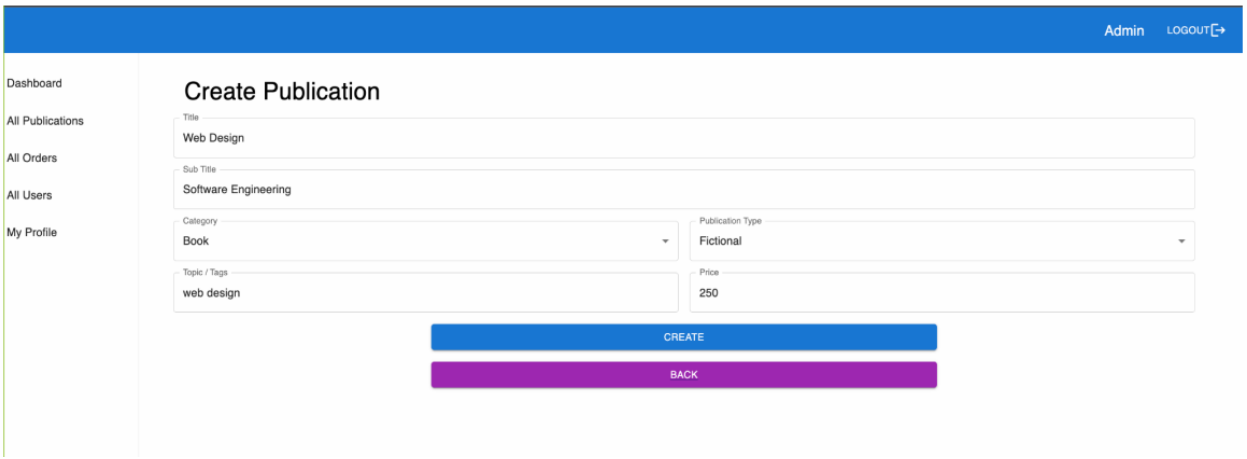

Рисунок 4.4 - Створення публікації з усіма обов'язковими полями.

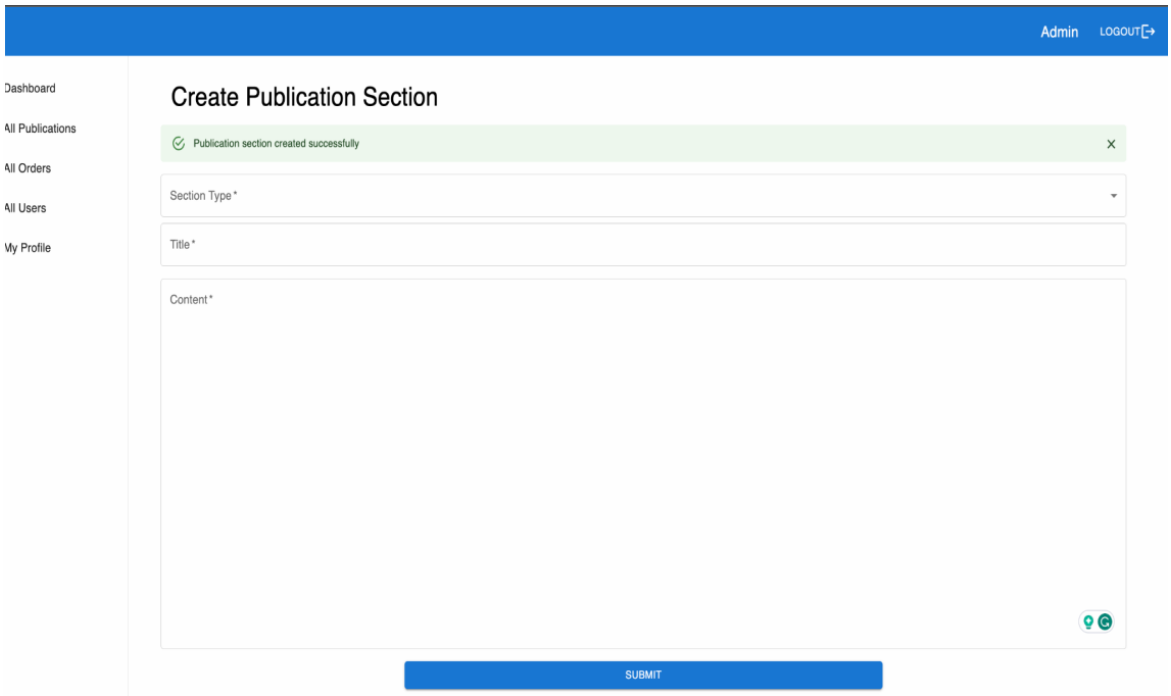

Рисунок 4.5 – Створена секція піблікації

Тест-кейс 2.2: Створення публікації з відсутніми обов'язковими полями.

Дії: Залишити одне або кілька обов'язкових полів форми створення публікації незаповненими та спробувати зберегти.

Очікуваний результат: Система відображає повідомлення про помилку валідації та запобігає створенню публікації.

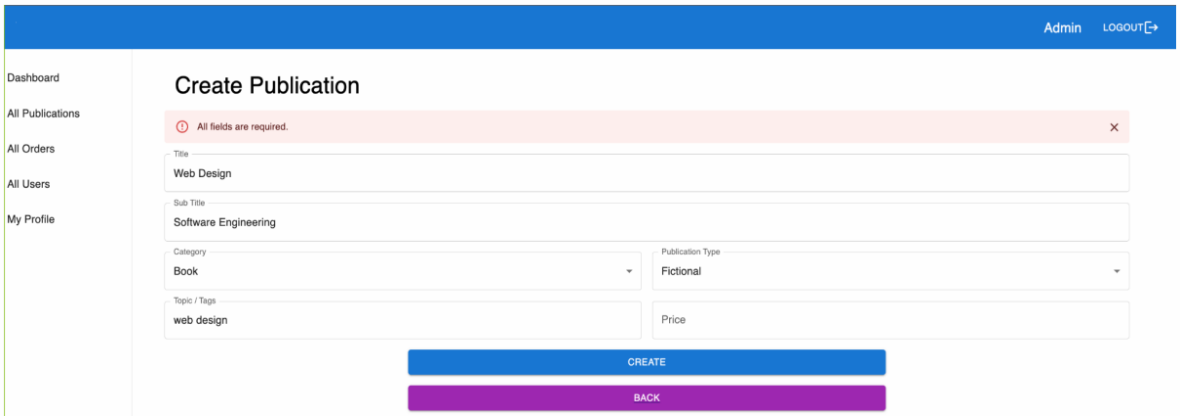

Рисунок 4.5 - Створення публікації з відсутніми обов'язковими полями

4.1.3. Функціональність створення розділу публікації

Тест-кейс 3.1: Створення розділу публікації з усіма обов'язковими полями.

Дії: Заповнити всі обов'язкові поля форми створення розділу публікації коректними даними та ініціювати збереження.

Очікуваний результат: Розділ публікації успішно створений та пов'язаний з відповідною публікацією.

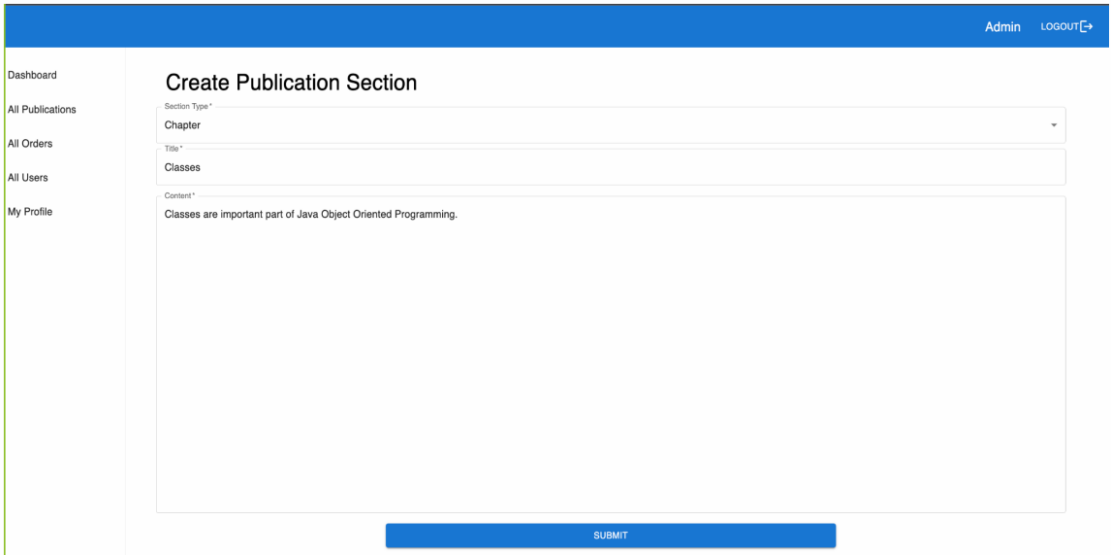

Рисунок 4.6 - Створення розділу публікації з усіма обов'язковими полями

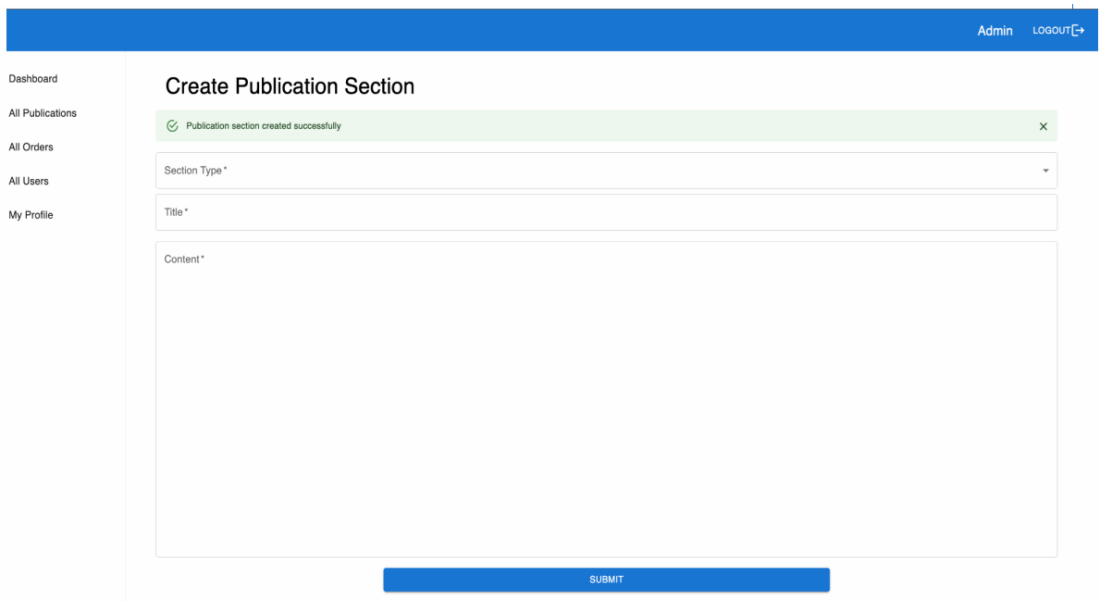

Рисунок 4.7 – Повідомлення про створену секцію

Тест-кейс 3.2: Створення розділу публікації з відсутніми обов'язковими полями.

Дії: Залишити одне або кілька обов'язкових полів форми створення розділу публікації незаповненими та спробувати зберегти.

Очікуваний результат: Система відображає повідомлення про помилку валідації та запобігає створенню розділу.

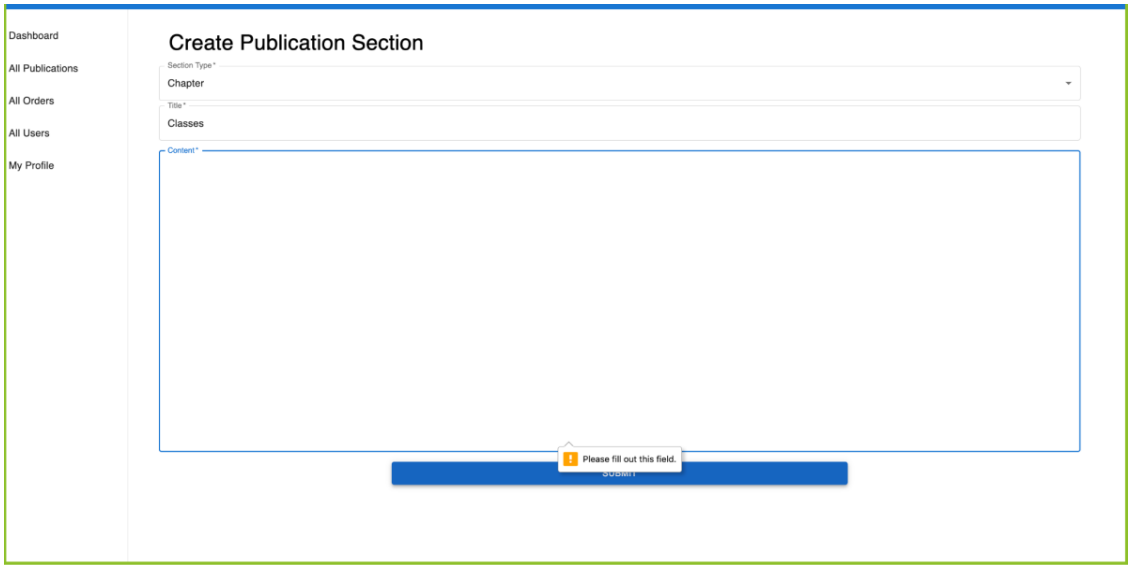

Рисунок 4.8 - Створення розділу публікації з відсутніми обов'язковими полями.

4.1.4. Функціональність редагування публікації

Тест-кейс 4.1: Редагування публікації.

Дії: Вибрати існуючу публікацію, внести зміни в її атрибути (наприклад, назву, ціну) та зберегти.

Очікуваний результат: Зміни успішно збережені, і публікація відображає оновлені дані.

Тест-кейс 4.2: Спроба редагування розділу публікації без відповідних дозволів.

Дії: Користувач з роллю, що не має прав на редагування розділів (наприклад, Дистриб'ютор), намагається модифікувати розділ публікації.

Очікуваний результат: Система відображає повідомлення про помилку доступу (наприклад, "Доступ заборонено") та запобігає модифікації.

Рисунок 4.9 - Спроба редагування розділу публікації без відповідних дозволів.

4.1.5. Функціональність перегляду всіх публікацій

Тест-кейс 5.1: Перегляд всіх публікацій.

Дії: Натиснути кнопку або посилання "Переглянути всі публікації".

Очікуваний результат: Система відображає повний список усіх доступних публікацій.

4.1.6. Функціональність створення замовлення

Тест-кейс 6.1: Створення замовлення з усіма обов'язковими полями.

Дії: Заповнити всі обов'язкові поля форми створення замовлення (наприклад, ID публікації, кількість) та ініціювати створення замовлення.

Очікуваний результат: Замовлення успішно створено, і його деталі відображаються в системі.

Тест-кейс 6.2: Створення замовлення з відсутніми обов'язковими полями.

Дії: Залишити одне або кілька обов'язкових полів форми створення замовлення незаповненими та спробувати створити замовлення.

Очікуваний результат: Система відображає повідомлення про помилку валідації та запобігає створенню замовлення.

4.1.7. Функціональність управління ролями користувачів

Тест-кейс 7.1: Додавання нового користувача.

Дії: Адміністратор заповнює форму додавання нового користувача з усіма необхідними даними, включаючи роль, та ініціює збереження.

Очікуваний результат: Новий користувач успішно доданий до системи та може автентифікуватися з призначеною роллю.

4.2. Техніки генерації тест-кейсів

Для забезпечення всебічного тестування системи були використані наступні техніки генерації тест-кейсів:

4.2.1. Аналіз граничних значень (Boundary Value Analysis, BVA)

Ця техніка полягає в тестуванні системи з вхідними значеннями, що розташовані безпосередньо на межах (мінімальні та максимальні) та за

межами допустимого діапазону. Це дозволяє виявити помилки, які часто виникають на "краях" вхідних даних.

Приклад: Перевірка реакції системи при створенні публікації з максимально допустимою кількістю розділів (наприклад, якщо існує обмеження на кількість розділів).

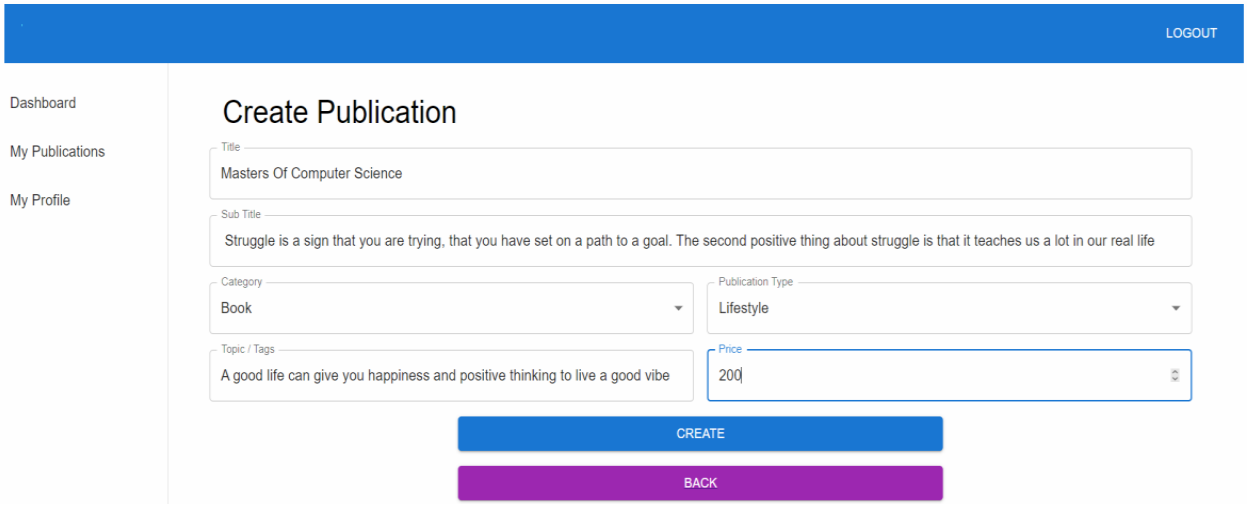

Рисунок 4.10 - Створення публікації з максимальною кількістю розділів

4.2.2. Розбиття на класи еквівалентності (Equivalence Partitioning, EP)

Ця техніка передбачає поділ області вхідних даних на класи еквівалентності, з яких обираються репрезентативні значення для тестування. Тестування одного значення з класу еквівалентності вважається достатнім для перевірки поведінки системи для всіх інших значень цього класу.

Приклад: Оцінка поведінки системи при створенні публікації з різними типами розділів (наприклад, "Вступ", "Основна частина", "Висновок"), якщо їх обробка може відрізнятися.

4.2.3. Тестування переходу станів (State Transition Testing)

Цей метод сфокусований на спостереженні за поведінкою системи під час переходу між різними станами. Він дозволяє виявити помилки, пов'язані

з некоректними змінами станів або несподіваними результатами при певних послідовностях дій.

Приклад: Оцінка поведінки системи, коли редактор модифікує частину розділу публікації, який вже був створений та знаходиться в певному стані (наприклад, "опублікований" або "на доопрацюванні").

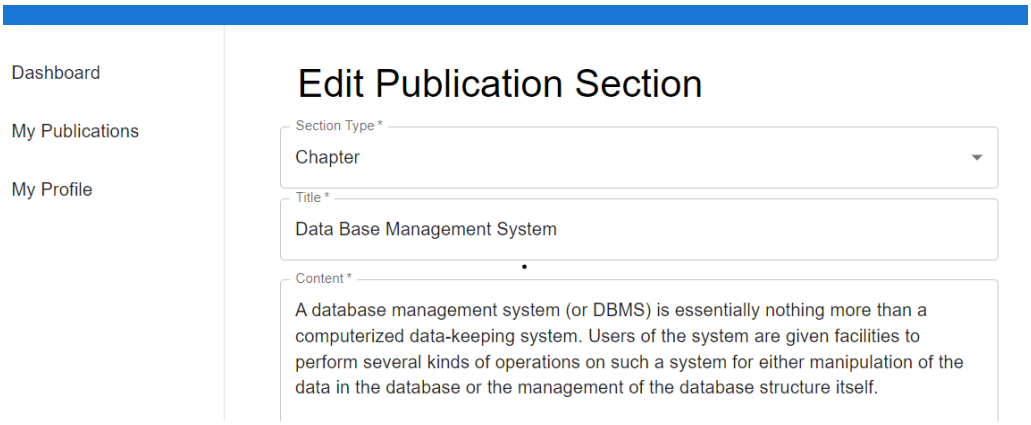

Рисунок 4.11 - Редактор змінює частину розділу публікації.

4.2.4. Тестування варіантів використання (Use Case Testing)

Ця техніка передбачає оцінку поведінки системи щодо заздалегідь визначених варіантів використання або сценаріїв, що відображають типові або критичні взаємодії користувачів із системою.

Приклад: Оцінка реакції системи, коли дистриб'ютор розміщує замовлення на книгу, ціна якої встановлена на 0 (нуль), перевіряючи коректність розрахунку загальної вартості та обробки такого замовлення.

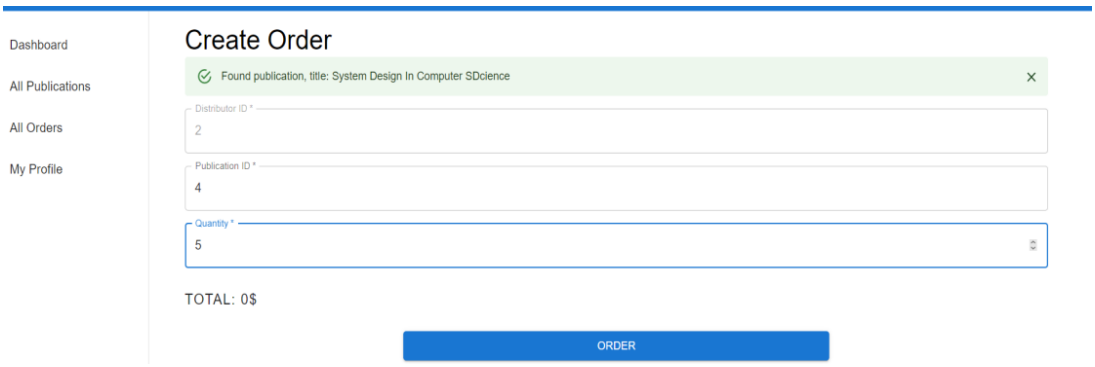

Рисунок 4.12 - Дистриб'ютор робить замовлення на книгу

Цей план тестування забезпечує структурований підхід до перевірки функціональності та надійності розроблюваної системи.

Розділ тестування, представлений у цьому документі, відображає систематичний підхід до забезпечення якості системи. На основі визначених вимог та специфікацій був розроблений детальний план тестування системного рівня, що охоплює ключові функціональні модулі та критичні сценарії взаємодії користувачів.

Застосування цих технік у поєднанні з ретельним розробленням тест-кейсів забезпечує високий рівень покриття та достовірності результатів тестування. Таким чином, представлений план тестування є надійною основою для верифікації відповідності системи автоматизації вимогам якості, функціональності та безпеки, що сприятиме впровадженню стабільного та ефективного програмного продукту.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено комплексне дослідження предметної області автоматизації процесів у видавничій сфері, розроблено архітектуру, реалізовано функціональні модулі та виконано тестування системи автоматизації видавничої діяльності.

На основі системного аналізу визначено актуальність автоматизації процесів управління у видавництвах. Було обґрунтовано необхідність впровадження сучасних інформаційних технологій для підвищення ефективності роботи з контентом, управління публікаціями, обробки замовлень та взаємодії з користувачами. Також визначено ключові функціональні та нефункціональні вимоги до системи, зокрема забезпечення масштабованості, безпеки даних, адаптивності та зручності для кінцевих користувачів.

Під час проєктування системи було враховано потреби основних зацікавлених сторін (стейкхолдерів), розроблено детальну специфікацію варіантів використання, побудовано діаграми варіантів використання, класів та послідовностей. Було обґрунтовано вибір архітектурного стилю Model-View-Controller (MVC), що забезпечує розділення відповідальностей між компонентами системи, спрощує супровід та розширення програмного забезпечення.

Розроблена система включає модулі управління аутентифікацією, контентом, замовленнями та користувачами. В межах реалізації були створені відповідні інтерфейси користувача та бекенд-логіка, що забезпечує обробку транзакцій, збереження даних та виконання основних бізнес-процесів. Система побудована з використанням сучасних технологій розробки веб-додатків, із застосуванням концепцій контролерів, сервісів та сутностей, що відображають логіку видавничої діяльності.

На етапі тестування реалізовано план системного тестування, побудовано тест-кейси, що охоплюють ключові функціональні сценарії. Застосовано різноманітні техніки тестування, зокрема аналіз граничних значень, розбиття на класи еквівалентності, тестування переходів станів і варіантів використання. Результати тестування підтвердили відповідність системи поставленим функціональним і нефункціональним вимогам.

Таким чином, у рамках дипломної роботи було повністю реалізовано цикл створення програмного продукту: від аналізу предметної області й постановки вимог до побудови архітектури, реалізації основних модулів та їх тестування. Запропоноване рішення є ефективним інструментом для автоматизації видавничої діяльності, здатним підвищити продуктивність редакційних процесів, забезпечити зручність керування публікаціями та замовленнями, а також знизити ризики, пов’язані з людським фактором.

У перспективі система може бути доопрацьована шляхом інтеграції з платіжними сервісами, впровадження аналітики користувацької активності, а також додавання інструментів для багатомовної підтримки контенту та адаптації до мобільних пристроїв.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бурковський В.В. Інформаційні системи в менеджменті. — Київ: КНЕУ, 2020. — 312 с.
2. Семенець О.І., Тарасов А.В. Основи видавничої справи. — Київ: Либідь, 2019. — 245 с.
3. Sommerville I. Software Engineering. 10th ed. — Pearson, 2016. — 792 p.
4. Шевченко А.М. Автоматизація документообігу в установах. — Київ: Видавництво НАУ, 2021. — 198 с.
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. — Addison-Wesley, 1994. — 395 p.
6. Грінченко С.В. Системний аналіз: підручник. — Київ: Центр учбової літератури, 2018. — 420 с.
7. IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. — IEEE, 1998.
8. Фіщук Н.П., Поліщук Ю.С. Інформаційні системи у видавничій діяльності. — Львів: Видавництво ЛНУ, 2020. — 175 с.
9. Pressman R.S. Software Engineering: A Practitioner’s Approach. 8th ed. — McGraw-Hill, 2014. — 928 p.
10. Троян В.В. Тестування програмного забезпечення. — Харків: ХНУРЕ, 2019. — 246 с.
11. Martin R.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. — Prentice Hall, 2017. — 432 p.
12. Larman C. Applying UML and Patterns. 3rd ed. — Pearson, 2004. — 736 p.
13. ISO/IEC/IEEE 29148:2018. Systems and Software Engineering — Life Cycle Processes — Requirements Engineering.
14. Fowler M. Patterns of Enterprise Application Architecture. — Addison-Wesley, 2002. — 560 p.

15. Alur D., Crupi J., Malks D. Core J2EE Patterns: Best Practices and Design Strategies. 2nd ed. — Prentice Hall, 2003. — 752 p.
16. Шаповал М.І. Програмна інженерія: основи проєктування. — Львів: Видавництво НУ “ЛП”, 2022. — 340 с.
17. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE).
18. Баскаков В.М. Проєктування програмних систем. — Харків: ХНЕУ, 2018. — 276 с.
19. Гладкий Ю.М. Моделювання та аналіз інформаційних систем. — Київ: КНЕУ, 2021. — 230 с.
20. Freeman E., Robson E. Head First Design Patterns. 2nd ed. — O’Reilly Media, 2020. — 694 p.
21. Малюков С.Л. Основи веб-програмування. — Київ: Видавництво КНЕУ, 2020. — 192 с.
22. ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes.
23. Beizer B. Software Testing Techniques. 2nd ed. — Van Nostrand Reinhold, 1990. — 550 p.
24. McConnell S. Code Complete. 2nd ed. — Microsoft Press, 2004. — 960 p.
25. Чорний А.О. UML: моделювання програмних систем. — Київ: Видавництво “Магістр”, 2021. — 210 с.
26. Pal K., Shrivastava P. Software Testing: Concepts and Practices. — Oxford University Press, 2021. — 300 p.
27. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. — Addison-Wesley, 2012. — 624 p.
28. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. — Addison-Wesley, 2005. — 496 p.
29. Tilley S., Huang S. Software Testing and Quality Assurance: Theory and Practice. — Wiley, 2011. — 400 p.

30. Sommerville I. Requirements Engineering: A Good Practice Guide. — Wiley, 1997. — 240 p.
31. Hohpe G., Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. — Addison-Wesley, 2003. — 735 p.
32. Murugesan S., Boehm B. Innovations in Systems and Software Engineering. — Springer, 2014. — 308 p.
33. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures. — University of California, 2000. — 180 p.
34. Kitchenham B. Evidence-Based Software Engineering and Systematic Reviews. — CRC Press, 2015. — 403 p.
35. MacKenzie I. Human-Computer Interaction: An Empirical Research Perspective. — Morgan Kaufmann, 2012. — 370 p.
36. ISO/IEC 9126-1:2001. Software engineering — Product quality — Part 1: Quality model.
37. Brooks F.P. The Mythical Man-Month: Essays on Software Engineering. Anniversary Ed. — Addison-Wesley, 1995. — 322 p.
38. Buschmann F., Henney K., Schmidt D.C. Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing. — Wiley, 2007. — 490 p.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Система автоматизації видавничої діяльності ”

Обсяг пояснювальної записки: 77 аркушів.

Дата закінчення роботи: 9 червня 2025 р.

Підпис студента _____