

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка системи керування каталогом коміксів Інтернет – магазину»

здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБз-19-1,
спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за
галузями)»


(підпис)

Полішученка Владислава Анаталійовича

Науковий керівник


(підпис)

Погореловська І. Д., к.е.н., доцент

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

Керівник освітньо-професійної
програми


(підпис)

Федотова-Півень І.М., к.т.н., доцент

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

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем, протокол від «01» червня 2023 № 14

Завідувач кафедри


(підпис)

Філоненко М. М., к.ф.-м.н., доцент

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

Ірпінь – 2023

Реєстрація № 40 07.06.2023
(дата)


(підпис відповідальної особи)

Результати перевірки до захисту 01.06.2023 
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 16.06.2023 
(оцінка) (дата) (підпис)

Голова комісії: 
(підпис)

В.М. Франчук

Члени комісії: 
(підпис)

М.М. Філоненко


(підпис)

В.О. Ніжегородцев


(підпис)

Є.В. Котух

БЛАНК ОЦІНЮВАННЯ ВИПУСКНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	29		16.06.2023 Франчук В.М. <i>[підпис]</i> Філоненко М.М. <i>[підпис]</i> Ніжегородцев В.О. <i>[підпис]</i>
2. Дотримання вимог щодо оформлення	від 0 до 10	8		Котух Є.В. <i>[підпис]</i> Суховерхівка Л.П. <i>[підпис]</i>
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником - 45 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
90	A	5 (відмінно)	16.06.2023	Франчук В.М. <i>[підпис]</i> Філоненко М.М. <i>[підпис]</i> Ніжегородцев В.О. <i>[підпис]</i> Котух Є.В. <i>[підпис]</i> Суховерхівка Л.П. <i>[підпис]</i>

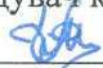
ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні управляючі системи та технології (за галузями)»

Затверджую

завідувач кафедри

 Філоненко М. М.
«16» 11 2022

**Завдання на кваліфікаційну роботу (проект)
здобувача вищої освіти
Поліщученка Владислава Анаталійовича**

1. Тема кваліфікаційної роботи: Розробка системи керування каталогом коміксів Інтернет – магазину
затверджена наказом ректора від 15 листопада 2022 № 1205 (зі змінами наказ від «04» квітня 2023 № 361)
2. Строк здачі закінченої роботи: 01.06.2023
3. Цільова установка: застосування набутих знань, умінь в розробці системи керування каталогом коміксів Інтернет – магазин
Мета роботи: створення системи керування каталогом коміксів
Об'єкт дослідження: електронні каталоги коміксів
Предмет дослідження: програмні засоби розробки систем керування каталогом коміксів
4. Вихідні дані до роботи: інформаційна модель керування каталогом коміксів Інтернет – магазину, технології розробки системи керування каталогом коміксів
5. Перелік графічного матеріалу: схеми етапів розробки, копії екранів роботи системи, лістинг програмного коду, презентація роботи (.ppt).
6. Зміст кваліфікаційної роботи: Розділ 1. Аналіз предметної області, 1.1. Огляд ринку коміксів та їх роль в сучасному світі, 1.2. Огляд існуючих систем керування каталогом коміксів, Розділ 2. Вимоги до системи керування каталогом коміксів, 2.1. Функціональні вимоги, 2.2 Нефункціональні вимоги, 2.3. Клієнт-серверна архітектура, 2.4 Веб-орієнтована архітектура, Розділ 3. Опис компонентів системи, 3.1 Клієнтська частина, 3.2 Серверна частина, 3.3. Процес додавання нового коміксу в каталог, 3.4. Організація пошуку коміксу в каталозі, 3.5 Процес оформлення замовлення на комікс. Розділ 4. Розгортання та налаштування системи, 4.1 Вимоги до апаратного та програмного забезпечення, 4.2 Розгортання системи, 4.3 Налаштування системи, 4.4. План тестування, 4.5. Результати тестування розробленого додатку.

7. Календарний план виконання роботи

№ п/п	Назва етапів випускної кваліфікаційної (дипломної) роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури ДР	До 15.11.2022	15.11.2022
2	1. Вступ та I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	06.02.2023	06.02.2023
3	II розділ (аналіз експериментальної частини і методики досліджень)	31.03.2023	31.03.2023
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	05.05.2023	05.05.2023
5	Висновки	16.05.2022	16.05.2022
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	01.06.2023	01.06.2023
7	Отримання відгуку керівника	08.06.2023	08.06.2023
8	Отримання зовнішньої рецензії	11.06.2023	11.06.2023

8. Дата видачі завдання «16» листопада 2022

9. Керівник дипломної роботи  Погорєловська І.Д., к.е.н., д

10. Гарант освітньо-професійної програми  Федотова-Півень І.М., к.т.н

11. Завдання до виконання отримав здобувач вищої освіти  Полішученко В.А.

АНОТАЦІЯ

«Розробка системи керування каталогом коміксів Інтернет – магазину»

Мета роботи: розробка системи управління каталогом коміксів для інтернет-магазину коміксів.

Завдання, що були вирішені при виконанні роботи:

проаналізовано функціоналу найбільш популярних систем керування каталогом коміксів, виявлення їх переваг та недоліків;

визначено особливості функціональних та нефункціональних вимог до систем керування каталогом коміксів;

здійснено вибір інструментів для розробки системи керування каталогом коміксів;

виконано розробку, розгортання та налаштування системи.

Предмет дослідження: програмні засоби розробки систем керування каталогом коміксів

Об'єкт дослідження: системи управління електронними каталогами коміксів.

Визначені у результаті роботи принципи розробки систем управління каталогами, функціональні можливості сучасних веб-технологій та методи тестування можуть знайти застосування у процесах покращення функціоналу та користувацького інтерфейсу систем управління каталогами коміксів.

Кваліфікаційна робота містить 80 сторінок, 8 таблиць, 8 рисунків, список використаних джерел 20 найменувань, 2 додатки

Ключові слова: Інтернет-магазин, HTML, CSS, React, Node.js, JavaScript

Рік виконання кваліфікаційної роботи: 2022-2023

Рік захисту роботи: 2023

ANNOTATION

«Development of a Catalog Management System for an Internet Comics Store»

The purpose of the work: development of a comic book catalog management system for an online comic book store.

The tasks that were solved during the performance of the work: the functionality of the most popular comics catalog management systems was analyzed, their advantages and disadvantages were identified; features of functional and non-functional requirements for comic book catalog management systems are defined; the selection of tools for the development of the comics catalog management system was made; the development, deployment and configuration of the system has been completed.

Research subject: software tools for developing comics catalog management systems.

Object of research: management systems of electronic catalogs of comics.

The principles of developing catalog management systems, the functionality of modern web technologies and testing methods determined as a result of the work can find application in the processes of improving the functionality and user interface of comic book catalog management systems.

The qualification work consists of 80 pages, 8 table, 8 figures, a list of 20 references, and 2 appendices.

Keywords: Internet store, HTML, CSS, React, Node.js, JavaScript

Year of qualification work: 2022-2023

Year of thesis defense: 2023

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	5
1.1 Огляд ринку коміксів та їх роль у сучасному світі	
1.2 Огляд існуючих систем керування каталогом коміксів	9
Висновок до розділу	10
РОЗДІЛ 2. ВИМОГИ ДО СИСТЕМИ КЕРУВАННЯ КАТАЛОГОМ КОМІКСІВ	12
2.1. Функціональні вимоги	12
2.2 Нефункціональні вимоги	14
2.3. Клієнт-серверна архітектура	16
2.4 Веб-орієнтована архітектура	19
Висновки до розділу	22
РОЗДІЛ 3. ОПИС КОМПОНЕНТІВ СИСТЕМИ	24
3.1 Клієнтська частина	24
3.2 Серверна частина	26
3.3. Процес додавання нового коміксу в каталог	39
3.4. Організація пошуку коміксу в каталозі	41
3.5 Процес оформлення замовлення на комікс	43
Висновки до розділу	47
РОЗДІЛ 4. РОЗГОРТАННЯ ТА НАЛАШТУВАННЯ СИСТЕМИ	48
4.1 Вимоги до апаратного та програмного забезпечення	48
4.2 Розгортання системи	50
4.3 Налаштування системи	53
4.4 План тестування	55
4.5. Результати тестування розробленого додатку	58
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	63
ДОДАТКИ	65
Додаток А	65
Додаток Б	70

ВСТУП

Цей проект спрямований на розробку системи управління каталогом коміксів для інтернет-магазину коміксів. Система дозволить інтернет-магазину ефективно управляти своїм запасом коміксів, полегшуючи користувачам пошук та купівлю коміксів. Система буде розроблена з використанням сучасних веб-технологій, що зробить її простою у використанні та доступною на різних пристроях.

Тема розробки системи управління каталогом коміксів для інтернет-магазину коміксів є дуже актуальною в сучасному світі. З появою нових технологій і зростанням популярності коміксів, індустрія коміксів переживає значний розвиток. Однак, багатьом інтернет-магазинам коміксів необхідна ефективна система управління каталогом для полегшення пошуку та покупки коміксів, що робить цю тему актуальною.

Стан і ступінь розробки проблеми в спеціальній літературі: За останні роки спостерігався зростаючий інтерес до цієї проблеми, оскільки комікси стали популярним видом розваги і залучають все більше шанувальників.

У деяких дослідженнях було звернуто увагу на необхідність розробки системи управління каталогом коміксів для полегшення пошуку та покупки. Такі дослідження аналізують проблеми, з якими стикаються інтернет-магазини коміксів, такі як недостатня ефективність пошуку, незручності при перегляді коміксів та обмежена доступність на різних пристроях.

Деякі дослідження також розглядають технологічні аспекти розробки систем управління каталогом коміксів. Вони зосереджуються на використанні сучасних веб-технологій та методів, таких як веб-розробка, бази даних та інші інструменти, що дозволяють створювати прості та зручні системи.

Однак, важливо відзначити, що не всі аспекти проблеми були повністю розроблені і досліджені. Наприклад, інтеграція системи управління каталогом коміксів з іншими системами, такими як система управління замовленнями або система відстеження доставки, може бути об'єктом подальших досліджень.

Метою дослідження є розробка системи управління каталогом коміксів для інтернет-магазину коміксів для полегшення користувачам пошуку та купівлі коміксів, а також для покращення управління каталогом у самому магазині.

Для досягнення мети було визначено наступні завдання:

- проаналізувати функціонал найбільш популярних систем керування каталогом коміксів з метою виявлення їх переваг та недоліків;
- визначити особливості функціональних та нефункціональних вимог до систем керування каталогом коміксів;
- здійснити вибір інструментів для розробки системи керування каталогом коміксів;
- виконати розробку, розгортання та налаштування системи.

Об'єктом дослідження є системи управління електронними каталогами коміксів.

Предмет дослідження: програмні засоби розробки систем керування каталогом коміксів

Методи дослідження, які були використані при виконанні роботи: системний аналіз - при роботі з інформаційними ресурсами за темою дослідження; методи порівняння та аналогій були застосовані при дослідженні функціональних можливостей найбільш популярних систем управління каталогами коміксів; методи групування – при проектуванні архітектури системи та схеми бази даних, розробці фронтенду та бекенду системи з використанням сучасних веб-технологій; тестування системи для перевірки відповідності вимогам клієнта і виявлення помилок; за допомогою методу узагальнення були сформувані висновки.

Науково-практичне значення одержаних результатів: визначені у результаті роботи принципи розробки систем управління каталогами, функціональні можливості сучасних веб-технологій та методи тестування можуть знайти застосування у процесах покращення функціоналу та

користувацького інтерфейсу систем управління каталогами коміксів, що позитивно вплине на розміри продаж, залучення нових клієнтів та утримання існуючих.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Протягом десятиліть комікси є невід'ємною частиною популярної культури, і фанати різного віку та походження насолоджуються історіями та ілюстраціями, представленими в цих виданнях. Індустрія коміксів зазнала значних змін за ці роки, з появою інтернет-магазинів коміксів та цифрових коміксів. Ці зміни призвели до нових викликів для магазинів коміксів, у тому числі до потреби в ефективних системах управління каталогами коміксів. У цьому розділі проаналізовано предметну область розробки системи управління каталогом коміксів для інтернет-магазину коміксів [1].

Інтернет-магазини коміксів стають все більш популярними в останні роки, надаючи шанувальникам коміксів легкий доступ до широкого спектру назв. Ці магазини пропонують зручний та ефективний спосіб придбання коміксів, причому багато з них пропонують як цифрові версії коміксів, так і фізичні копії. Однак багатьом інтернет-магазинам коміксів бракує ефективної системи управління каталогом коміксів, що може призвести до незадовільного користувацького досвіду та втрат продажів.

Система управління каталогом коміксів є важливим компонентом інтернет-магазину коміксів. Вона дозволяє власникам магазинів ефективно управляти своїми запасами, полегшуючи користувачам пошук і купівлю коміксів. Ефективна система управління каталогом повинна дозволяти власникам магазинів додавати нові видання, оновлювати наявні та видаляти видання, які більше не доступні. Вона також повинна дозволяти користувачам шукати комікси за різними критеріями пошуку, такими як назва, автор, видавництво та жанр. Крім того, вона повинна надавати опції сортування та фільтрації, щоб допомогти користувачам швидко і легко знаходити те, що вони шукають.

1.1 Огляд ринку коміксів та їх роль в сучасному світі

Комікси - це вид мистецтва, який поєднує розповідь з візуальним мистецтвом. Вони існують вже багато десятиліть і перетворилися на популярну

форму розваг у всьому світі. Ринок коміксів значно зріс за останні роки і продовжує приваблювати нових читачів різного віку. У цьому есе я надам огляд ринку коміксів та його ролі в сучасному суспільстві.

Ринок коміксів поділяється на кілька категорій, включаючи комікси про супергероїв, мангу, графічні романи та інді-комікси. Комікси про супергероїв є найвідомішим і найпопулярнішим видом коміксів, з такими персонажами, як Бетмен, Супермен, Людина-павук і Диво-жінка, які є одними з найбільш впізнаваних. Манга, з іншого боку, - це тип коміксів, що походить з Японії і відомий своїм унікальним стилем і технікою розповіді. Графічні романи - це довші, складніші історії, розказані у форматі коміксів, які часто стосуються більш серйозних тем, ніж традиційні комікси. Інді-комікси, також відомі як альтернативні комікси, видаються самвидавом або невеликими незалежними видавництвами і часто мають більш нішеву аудиторію [3].

Ринок коміксів значно зріс за останні роки: у 2018 році продажі сягнули понад 1 мільярд доларів. Частково це зростання зумовлене популярністю фільмів про супергероїв та телевізійних шоу, які познайомили з персонажами нову аудиторію. Крім того, поява цифрових коміксів полегшила читачам доступ до коміксів та їхню купівлю в Інтернеті. Багато видавців тепер пропонують цифрові версії своїх коміксів, які можна завантажити на планшети чи смартфони для зручного читання.

Комікси також відіграли значну роль у популярній культурі, а супергерої стали одними з найбільш впізнаваних персонажів у світі. Від одягу та іграшок до фільмів і телевізійних шоу, супергерої стали основною частиною мейнстрімних розваг. Це призвело до більшого визнання і поцінування коміксів як легітимної форми мистецтва і розповіді історій.

Однією з найважливіших подій на ринку коміксів стала поява графічних романів. Графічні романи - це довші, складніші історії, які часто пишуться для дорослих і стосуються більш серйозних тем. Вони набули популярності в останні роки і стали важливою частиною сучасної літератури. Багато графічних

романів стосуються соціальних і політичних питань, таких як раса, гендер і сексуальність, і допомогли підвищити обізнаність і розуміння цих питань.[5]

Комікси також відіграють важливу роль в освіті. Багато вчителів та бібліотекарів використовують комікси як спосіб залучити учнів до читання та навчання. Комікси можуть бути ефективним інструментом для навчання навичкам грамотності, оскільки вони вимагають від читачів інтерпретації як тексту, так і зображень. Крім того, комікси можна використовувати для викладання історії, природничих наук та інших предметів, оскільки вони можуть подавати складну інформацію в цікавій і доступній формі.

Комікси також стали потужним інструментом соціального коментування. Багато коміксів стосуються важливих соціальних питань, таких як расизм, сексизм і гомофобія, і можуть стати платформою для дискусій і дебатів. Комікси також використовуються для підвищення обізнаності та збору коштів на соціальні цілі, такі як захист довкілля, прав людини та ліквідація наслідків стихійних лих.

Ринок коміксів значно зріс за останні роки і став важливою частиною сучасної культури. Комікси відіграють важливу роль у масовій культурі, освіті та соціальному коментуванні. Оскільки ринок продовжує розвиватися, комікси, ймовірно, і надалі відіграватимуть важливу роль у формуванні нашого розуміння навколишнього світу.

Комікс-індустрія континентальної Європи та Великобританії самотня, індивідуальна і відрізняється від американської традиції. Європа не настільки прив'язана до комерційно безпрограшних фантастичних сюжетів. На відміну від Америки, європейці вкрай неохоче звертаються, наприклад, до героїчних історій, зате займають провідну позицію (як кількісно, так і якісно) у таких жанрах, як соціально-побутовий роман, вестерн, гумористика. Більшою мірою європейці тяжіли до реалізму, авангарду, комізму, орієнтуючись на читача більш затребуваного, інтелектуально підготовленого (якщо говорити про читача дорослого). Інколи графічну прозу в Європі навіть видають інакше – у вигляді дорогих альбомів у твердих палітурках (хоча, звичайно, існує комікс і в

газетно-журнальному варіанті). У Франції, Бельгії та Іспанії комікси вважаються надбанням національної культури і користуються офіційною (і фінансовою в тому числі) підтримкою. Наприклад, в Іспанії каталонський центр ЮНЕСКО видав об'ємний каталог творів вітчизняних комікс-авторів (художників) з докладним мистецтвознавчим аналізом їхньої творчості. З 1960-х років у Франції функціонує науково-дослідний інститут космічного жанру образотворчого мистецтва – Центр вивчення графічної літератури, а кращі роботи виставляються в Луврі. Щорічно проводиться безліч державних та міжнародних виставок та фестивалів, з яких найбільш відомим і престижним вважається фестиваль у Ангулемі (Франція).

Ще одна принципова відмінність від американського коміксу: виробництво коміксів в Європі не анонімно-конвеєрне (коли, приміром, над створенням чергових випусків «Людини-павука» працював цілий штат найманих художників і сценаристів), а поштучне, авторське.

Якщо в першій половині XX століття європейський комікс фантастичної тематики випускався рідко, й почасти залежав від американського (більшість супергероїв були дублетами американських), то вже у 1960-і роки в Європі з'явилася своя фантастика – вишукана та інтелектуальна.[5]

У 1962 році на сторінках журналу «V Magazine» вперше з'явився роман Жана-Клода Форе «Барбарелла» – шедевр комерційного арт-комікса, що майже відразу ж вийшов окремим альбомом у видавництві «Eric Losfeld».

«Барбарелла» – це іронічний і пародійний графічний роман у жанрі космічної опери, що тонко обігрує штампи жанру. Крім того, це перший еротичний і суто «дорослий» комікс у Європі.

А роком пізніше Європу підкорив мега-бестселер Рене Госінні й Альбера Удерзо «Астерікс» – гумористичних цикл, що нараховує більше 20 альбомів.

Справжньою Меккою європейського жанрового коміксу став журнал «Metal Hurlant», який за десятиліття свого існування (1975 – 1987) зробив справжню революцію у світі мальованої літератури. На тлі американських палп-коміксів (масових й дешевих), широкі за жанровим діапазоном роботи, що

друкувались у журналі, виділялися яскраво вираженою індивідуальністю графіки, витонченістю сюжетів і ретельним опрацюванням характерів. А головне – орієнтацією на молодих інтелектуалів.

Сучасна історія європейського коміксу це не стільки список серіалів і графічних романів, скільки «пантеон» іменитих художників і письменників, чия популярність простирається далеко за межами жанру мальованої літератури і чий авторитет у світі сучасного мистецтва не підлягає сумніву. Професіонали, що працюють над графічними історіями – не тільки художники. Так для коміксів Жанра Жиро сценарії писали великий режисер Федеріко Фелліні та кіноавангардист, містик і резонер Алессандро Ходоровський. А сам Жиро був ще й дизайнером і розкадровщиком: він придумав предметний світ для фільмів «Дюна», «Чужий», «Трон», «Віллоу», повнометражних мультфільмів «Володар часу» та «Малюк Немо», а також на його рахунку безліч книжкових ілюстрацій і дизайнерських розробок [6].

1.2 Огляд існуючих систем керування каталогом коміксів

Існує кілька систем для управління каталогами коміксів, починаючи від спеціалізованого програмного забезпечення і закінчуючи онлайн базами даних. У цьому огляді ми розглянемо деякі з найпопулярніших доступних варіантів.

ComicRack - це популярне програмне забезпечення для управління коміксами, яке дозволяє користувачам організовувати свої цифрові колекції коміксів на своїх комп'ютерах. Програма підтримує широкий спектр форматів файлів коміксів і включає такі функції, як тегування, оцінка та сортування.

Collectorz - ще одне популярне програмне забезпечення для управління коміксами, яке дозволяє користувачам організовувати та каталогізувати свої колекції коміксів. Програма підтримує широкий спектр платформ, включаючи Windows, Mac, iOS та Android. Collectorz також включає такі функції, як сканування штрих-коду, тегування та рейтинг.

CLZ Comics - це мобільний додаток, який дозволяє користувачам сканувати штрих-код своїх фізичних коміксів і додавати їх до свого цифрового

каталогу. Додаток включає такі функції, як синхронізація на різних пристроях, тегування та оцінка.

Comic Book DB - це онлайн-база даних коміксів, яка дозволяє користувачам шукати та відстежувати свої улюблені видання. Веб-сайт містить такі функції, як відгуки користувачів, рейтинги та форум спільноти.

MyComicShop - це інтернет-магазин, який спеціалізується на продажу коміксів, але також включає функцію каталогізації для користувачів, щоб організувати свої колекції. На сайті є такі функції, як пошук за назвою, видавництвом і персонажем, а також функція створення списку бажань.

League of Comic Geeks (Ліга коміксів) - це онлайн-спільнота шанувальників коміксів, яка включає функцію каталогізації, що дозволяє користувачам відстежувати свої колекції. Веб-сайт містить такі функції, як список бажань, список бажань та відгуки користувачів.

Comic Vine - ще одна онлайн-база даних коміксів, яка включає такі функції, як пошук за назвою, персонажем і видавцем. Веб-сайт також містить відгуки та рейтинги користувачів, а також форум спільноти.

Для управління каталогами коміксів існує широкий спектр можливостей, від спеціалізованого програмного забезпечення до онлайн-баз даних. Користувачі повинні враховувати свої конкретні потреби та вподобання при виборі системи для управління своїми колекціями.

Висновок до розділу

У результаті аналізу предметної області, можна зробити висновок про значущість коміксів у сучасному світі. Комікси стали популярним видом розваги та мистецтва, що залучає увагу різних верств населення. Ринок коміксів зростає, що свідчить про постійне збільшення попиту на них.

Також виявлено, що існують різні системи керування каталогом коміксів, які дозволяють легко знаходити потрібну інформацію про комікси та організовувати їх зберігання. Однак, на даний момент існуючі системи мають свої обмеження та необхідність у поліпшенні та розширенні функціоналу.

Розвиток ринку коміксів та систем керування каталогом коміксів є актуальними темами для дослідження та подальшого розвитку. Результати аналізу можуть бути корисними для компаній, які займаються виробництвом та продажем коміксів, а також для розробників систем керування каталогом коміксів.

РОЗДІЛ 2. ВИМОГИ ДО СИСТЕМИ КЕРУВАННЯ КАТАЛОГОМ КОМІКСІВ

2.1. Функціональні вимоги

Веб-система - це програмне забезпечення, доступне через веб-браузер в Інтернеті. Термін "веб-система" часто використовується як взаємозамінний з термінами "веб-додаток" або "веб-ресурс"[3].

Веб-системи призначені для забезпечення зручного інтерфейсу, до якого можна отримати доступ з будь-якої точки світу, якщо у користувача є підключення до Інтернету та веб-браузер. Вони часто використовуються для автоматизації бізнес-процесів, управління даними або надання онлайн-послуг.

Однією з головних переваг веб-систем є їхня доступність. Користувачі можуть отримати доступ до програми з будь-якого пристрою з веб-браузером, включаючи стаціонарні комп'ютери, ноутбуки, планшети та смартфони. Це робить веб-системи ідеальним вибором для компаній з розподіленою робочою силою або клієнтів, яким потрібно отримати доступ до програми з різних місць.

Ще однією перевагою веб-систем є те, що їх легко підтримувати та оновлювати. Оскільки додаток розміщується на веб-сервері, оновлення та обслуговування можуть виконуватися централізовано, без необхідності завантаження та встановлення нового програмного забезпечення користувачами. Це гарантує, що всі користувачі мають доступ до останньої версії програми, і що будь-які помилки або проблеми з безпекою можуть бути швидко вирішені.

Веб-системи, як правило, будуються з використанням комбінації інтерфейсних і бек-енд технологій. Фронтенд - це частина системи, з якою взаємодіє користувач, що зазвичай складається з інтерфейсу користувача (UI), який створюється за допомогою HTML, CSS і JavaScript. Back-end відповідає за обробку запитів від front-end і зазвичай складається з серверної мови програмування, такої як PHP, Python або Ruby.

Існує безліч веб-систем, починаючи від простих систем управління контентом (CMS) і закінчуючи складними системами планування ресурсів підприємства (ERP). CMS-системи використовуються для управління контентом веб-сайтів, тоді як ERP-системи використовуються для управління бізнес-процесами, такими як фінанси, управління персоналом та управління ланцюжками поставок.

Веб-системи можуть бути локальними або хмарними. Локальні системи розміщуються на серверах, розташованих у власному центрі обробки даних організації, тоді як хмарні системи розміщуються у сторонніх провайдерів, таких як Amazon Web Services, Microsoft Azure або Google Cloud [8].

Система управління каталогом коміксів - це інструмент, який дозволяє користувачам легко відстежувати свої колекції коміксів. Щоб переконатися, що система відповідає потребам користувачів, важливо встановити чіткі функціональні вимоги. У цьому розділі ми окреслимо деякі з ключових функціональних вимог, яким повинна відповідати система управління каталогом коміксів.

Однією з перших вимог будь-якої веб-системи є реєстрація та аутентифікація користувачів. Система управління каталогом коміксів повинна дозволяти користувачам реєструватися і створювати власні унікальні облікові записи. Це дозволить користувачам персоналізувати свій досвід і отримати доступ до своєї колекції коміксів з будь-якого пристрою з доступом до Інтернету. Система також повинна забезпечувати безпечну автентифікацію, щоб тільки авторизовані користувачі могли отримати доступ до своїх облікових записів [2].

Система управління каталогом коміксів повинна дозволяти користувачам додавати нові комікси до своєї колекції. Користувачі повинні мати можливість вводити таку інформацію, як назва, автор, художник, видавець, дата випуску та номер випуску коміксу. Система також повинна дозволяти користувачам редагувати цю інформацію, якщо вони помилилися або якщо якісь деталі змінилися з часом.

Однією з найважливіших функцій системи управління каталогом коміксів є пошук і фільтрація. Користувачі повинні мати можливість шукати свою колекцію за назвою, автором, художником, видавництвом, датою випуску та номером випуску. Система також повинна дозволяти користувачам фільтрувати свою колекцію за різними критеріями, такими як жанр, серія, персонаж тощо. Це дозволить користувачам швидко знайти комікс, який вони шукають.

Система управління каталогом коміксів повинна також дозволяти користувачам читати і попередньо переглядати комікси. Система повинна надавати користувачам можливість завантажувати цифрові версії своїх коміксів або посилання на цифрові версії, доступні в Інтернеті. Система також повинна надавати користувачам можливість попереднього перегляду коміксів перед читанням, наприклад, попередній перегляд обкладинки або попередній перегляд перших кількох сторінок [9].

Система управління каталогом коміксів повинна також дозволяти користувачам ділитися своїми колекціями з іншими. Система повинна надавати такі соціальні функції, як можливість стежити за іншими користувачами і переглядати їхні колекції, а також можливість ділитися окремими коміксами в соціальних мережах. Це дозволить користувачам спілкуватися з іншими людьми, які поділяють їхні інтереси, і розширювати свою колекцію.

Насамкінець, система управління каталогом коміксів повинна мати зручний інтерфейс, щоб користувачі могли легко керувати своїми колекціями. Система повинна включати такі функції, як реєстрація та автентифікація користувачів, введення та редагування коміксів, пошук та фільтрація, читання та попередній перегляд, а також обмін та соціальні функції. Відповідаючи цим функціональним вимогам, система управління каталогом коміксів може забезпечити користувачам комплексний і приємний досвід роботи.

2.2 Нефункціональні вимоги

Нефункціональні вимоги - це якості, якими повинна володіти система, щоб відповідати певним стандартам і очікуванням. Ці вимоги часто стосуються

продуктивності, надійності, безпеки, зручності та масштабованості. У цьому розділі ми опишемо деякі з нефункціональних вимог, які є важливими для системи управління каталогом коміксів.

Продуктивність: Система управління каталогом коміксів повинна бути здатна обробляти велику кількість даних і запитів користувачів без уповільнення або збоїв. Система повинна бути спроектована таким чином, щоб оптимізувати продуктивність, наприклад, за рахунок використання методів кешування або оптимізації запитів до бази даних [15].

Надійність: Система повинна бути надійною і доступною в будь-який час, з мінімальним часом простою для обслуговування або оновлення. Система має бути відмовостійкою, наприклад, за рахунок використання резервних серверів або методів балансування навантаження.

Безпека: Система повинна бути безпечною і захищати дані як користувачів, так і організації. Система повинна бути розроблена з декількома рівнями безпеки, наприклад, за допомогою шифрування, брандмауерів або списків контролю доступу. Система також повинна регулярно перевірятися на вразливості та оновлюватися найновішими патчами безпеки.

Зручність використання: Система повинна бути простою у використанні та зрозумілою як для початківців, так і для досвідчених користувачів. Система повинна мати зручний інтерфейс, чітку навігацію та корисний зворотній зв'язок. Система також повинна бути розроблена з урахуванням потреб людей з обмеженими можливостями, наприклад, з підтримкою програм для читання з екрану або альтернативних методів введення.

Масштабованість: Система повинна бути масштабованою, щоб відповідати зростанню кількості даних та користувачів. Система повинна бути спроектована для горизонтального масштабування, наприклад, шляхом додавання додаткових серверів для обробки збільшеного трафіку. Система також повинна бути спроектована для вертикального масштабування, наприклад, шляхом оптимізації бази даних або модернізації апаратних компонентів [14].

Інтероперабельність: Система повинна мати можливість інтегруватися з іншими системами та платформами. Система повинна бути розроблена з використанням стандартних протоколів і форматів, таких як REST API або XML. Система також повинна бути розроблена з урахуванням модульності, наприклад, з використанням мікросервісів або контейнеризації.

Зручність обслуговування: Систему має бути легко підтримувати та оновлювати з часом. Система повинна бути розроблена з чіткою документацією, коментарями до коду та послідовним стилем кодування. Система також повинна бути розроблена з автоматизованими процесами тестування та розгортання, такими як безперервна інтеграція та доставка.

Насамкінець, нефункціональні вимоги є важливими для забезпечення того, щоб система управління каталогом коміксів була продуктивною, надійною, безпечною, зручною, масштабованою, інтероперабельною та підтримуваною. Ці вимоги необхідно враховувати протягом усього процесу проектування та розробки, щоб система відповідала потребам як користувачів, так і організації.

2.3. Клієнт-серверна архітектура

Архітектура клієнт-сервер (рис. 2.1) є широко використовуваною моделлю для проектування та реалізації програмних додатків, в тому числі веб-і мобільних додатків. У цій архітектурі система поділяється на дві основні частини: клієнт і сервер. Клієнт - це користувацький інтерфейс, який запускається на пристрої користувача, наприклад, у веб-браузері або мобільному додатку. Сервер - це внутрішній компонент, який відповідає за обробку та зберігання даних, а також зв'язок з клієнтом.

Архітектура клієнт-сервер надає багато переваг, таких як масштабованість, безпека та гнучкість. Клієнтом може бути будь-який пристрій або додаток, який може взаємодіяти з сервером, а сервером може бути одна машина або кластер машин, які працюють разом для обробки великих обсягів

трафіку і даних. Сервер може бути розташований де завгодно, наприклад, локально, в хмарі або в гібридному середовищі.

Клієнт і сервер спілкуються один з одним за допомогою стандартизованого протоколу, такого як HTTP або TCP/IP. Клієнт надсилає запити на сервер, а сервер відповідає даними або діями на основі запиту. Сервер також може надсилати клієнту дані, наприклад, оновлення або сповіщення в режимі реального часу [20].

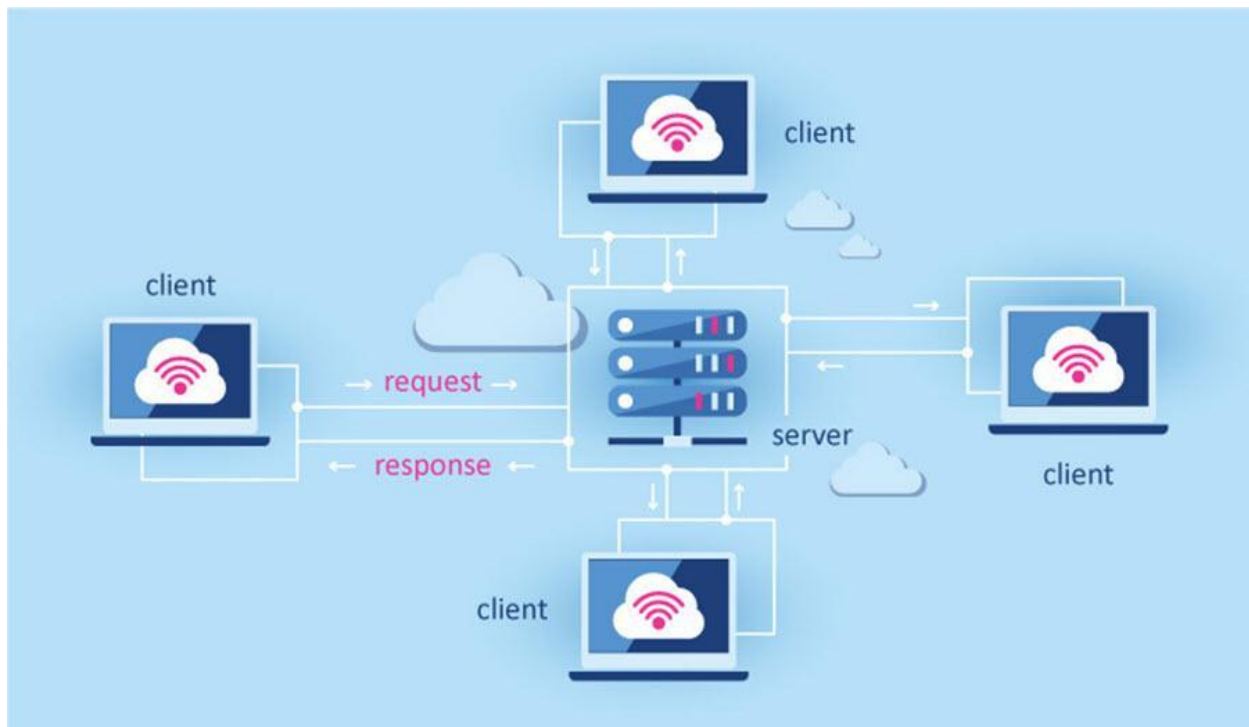


Рис. 2.1. Модель архітектури клієнт-сервер

Однією з ключових переваг клієнт-серверної архітектури є те, що вона дозволяє розділити обов'язки. Клієнт відповідає за роботу з користувацьким інтерфейсом та взаємодію з користувачем, тоді як сервер відповідає за обробку та зберігання даних. Такий поділ забезпечує кращу масштабованість і продуктивність, оскільки кожен компонент може бути оптимізований незалежно.

Клієнт-серверна архітектура також забезпечує кращу безпеку, оскільки сервер може контролювати доступ до даних і ресурсів. Сервер може реалізовувати механізми автентифікації та авторизації, такі як рівні входу та дозволів, щоб гарантувати, що лише авторизовані користувачі можуть отримати доступ до певних даних або виконувати певні дії. Клієнт-серверна

архітектура часто використовується у веб-додатках, де клієнтом є веб-браузер, а сервером - веб-сервер, на якому розміщується додаток. Клієнт надсилає запити до сервера за допомогою HTTP, а сервер відповідає кодом HTML, CSS і JavaScript, який браузер може відобразити і вивести на екран.

Клієнт і сервер можуть спілкуватися за допомогою різних типів протоколів, залежно від вимог програми. Наприклад, чат-додаток може використовувати WebSockets для спілкування в режимі реального часу, тоді як програма для обміну файлами може використовувати FTP або SFTP для передачі файлів [12].

Однією з переваг архітектури клієнт-сервер є те, що вона дозволяє модульний дизайн і розробку. Клієнт і сервер можуть розроблятися незалежно, а над кожним компонентом можуть працювати різні команди. Це може призвести до скорочення часу розробки та покращення якості коду.

Ще однією перевагою клієнт-серверної архітектури є те, що вона полегшує обслуговування та оновлення. Оскільки клієнт і сервер є окремими, оновлення та виправлення можуть бути застосовані до одного компонента, не впливаючи на інший. Це допомагає мінімізувати час простою і збоїв в роботі користувачів [22].

Клієнт-серверна архітектура також може допомогти підвищити продуктивність і масштабованість. Оскільки сервер відповідає за обробку та зберігання даних, його можна оптимізувати для обробки великих обсягів трафіку та даних. Це може допомогти гарантувати, що додаток залишатиметься швидким і чуйним навіть під час великих навантажень.

Однак у клієнт-серверної архітектури є і деякі потенційні недоліки. Один з них полягає в тому, що вона може призвести до затримок і мережних накладних витрат, оскільки запити і відповіді повинні передаватися через мережу. Це може бути особливо проблематично для додатків, які вимагають взаємодії в реальному часі або з низькою затримкою.

Ще однією потенційною проблемою є те, що архітектура клієнт-сервер може створювати складність, особливо якщо додаток вимагає складної обробки

даних або численних взаємодій між клієнтом і сервером. У деяких випадках доцільніше використовувати інші архітектури, наприклад, однорангові або розподілені системи.

Загалом, архітектура клієнт-сервер є гнучкою та потужною моделлю для проектування та реалізації програмних додатків. Розділяючи клієнтські та серверні компоненти, вона забезпечує модульний дизайн, полегшує обслуговування, підвищує продуктивність і масштабованість. Однак важливо ретельно зважити вимоги програми та потенційні недоліки перед тим, як обрати цю архітектуру.

2.4 Веб-орієнтована архітектура

Веб-орієнтована архітектура (WOA) (рис. 2.2) - це архітектурний стиль для проектування веб-систем і додатків. Він базується на принципах Всесвітньої павутини і наголошує на використанні стандартних веб-протоколів і технологій для побудови розподілених, масштабованих та інтероперабельних систем. Ось деякі ключові особливості та переваги WOA:

Базується на ресурсах: У WOA все є ресурсом, який можна ідентифікувати за допомогою URI (Uniform Resource Identifier - уніфікований ідентифікатор ресурсу). Ресурсами можуть бути документи, зображення, відео, послуги або будь-який інший тип даних. Доступ до них зазвичай здійснюється за допомогою методів HTTP, таких як GET, POST, PUT і DELETE.

Без громадянства: WOA розроблений як бездержавний, що означає, що кожен запит повинен містити всю інформацію, необхідну для його обробки, не покладаючись на будь-які попередні взаємодії або контекст. Це робить систему більш масштабованою, оскільки вона може обробляти велику кількість паралельних запитів без необхідності зберігати інформацію про стан для кожного з них.

Багаторівнева: WOA підтримує багаторівневу архітектуру, де кожен рівень виконує певну функцію і взаємодіє лише з сусідніми рівнями. Це

забезпечує більш модульний дизайн і полегшує додавання або видалення компонентів за потреби.

Інтероперабельність: WOA розроблений таким чином, щоб бути інтероперабельним, що означає, що він може працювати з різними технологіями, платформами та системами. Це досягається завдяки використанню стандартних веб-протоколів і форматів, таких як HTTP, XML, JSON і REST (Representational State Transfer) [20].

Масштабованість: WOA має високу масштабованість, оскільки дозволяє використовувати розподілені системи та механізми кешування для підвищення продуктивності та зменшення мережевого трафіку. Це робить його придатним для створення великомасштабних додатків, які можуть обробляти мільйони користувачів і транзакцій.

Безпека: WOA включає вбудовані механізми безпеки, такі як автентифікація, шифрування та контроль доступу, які допомагають захистити конфіденційні дані та запобігти несанкціонованому доступу.

Відпочиваючий: WOA часто використовується в поєднанні зі стилем веб-архітектури під назвою REST (Representational State Transfer - передача представницького стану), який підкреслює використання простих, заснованих на ресурсах інтерфейсів і стандартних методів HTTP. RESTful API (інтерфейси прикладного програмування) можуть використовуватися для розкриття ресурсів і функціональності у веб-системі, дозволяючи розробникам створювати клієнтські програми, які можуть взаємодіяти з системою через Інтернет.

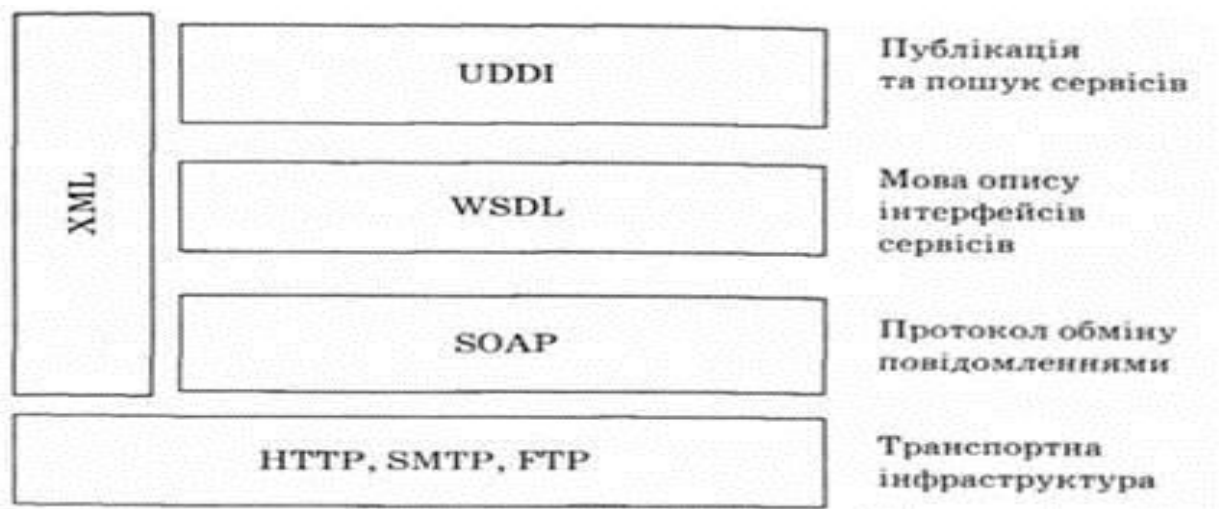


Рис. 2.2. Веб-орієнтована архітектура

Сервіс-орієнтований: WOA також можна вважати формою сервіс-орієнтованої архітектури (SOA), яка є способом проектування систем як набору незалежних сервісів, що взаємодіють один з одним через мережу. У веб-системі кожен ресурс можна розглядати як сервіс, який надає певну функцію або дані. [19]

Мікросервіси: WOA також можна використовувати в поєднанні з архітектурою мікросервісів, яка є способом проектування систем як набору невеликих, незалежно розгорнутих сервісів, які працюють разом, щоб забезпечити загальну функціональність системи. Кожен мікросервіс може бути реалізований з використанням принципів WOA, що забезпечує високу масштабованість і модульний дизайн.

HATEOAS: WOA також підтримує принцип HATEOAS (Hypermedia as the Engine of Application State), який означає, що кожен ресурс повинен містити посилання на інші пов'язані ресурси і дії, які можуть бути виконані над ними. Це дозволяє створити більш зрозумілу і самоописувану систему, де клієнти можуть переміщатися по доступним ресурсам і функціональності, використовуючи надані посилання.

REST vs SOAP: Розглядаючи використання WOA, важливо відзначити, що він часто використовується на противагу іншому стилю веб-архітектури, який називається SOAP (Simple Object Access Protocol - простий протокол доступу до об'єктів), який підкреслює використання повідомлень на основі

XML і набору заздалегідь визначених методів для комунікації. RESTful системи, як правило, простіші і легші, ніж системи на основі SOAP, але можуть підходити не для всіх типів додатків.

Отже, веб-орієнтована архітектура - це потужний і гнучкий підхід до проектування веб-систем і додатків. Завдяки використанню стандартних веб-протоколів і технологій, вона забезпечує інтероперабельність, масштабованість і модульний дизайн. Вона може використовуватися в поєднанні з іншими архітектурними стилями, такими як REST, SOA і мікросервіси, і підтримує такі принципи, як HATEOAS, для створення систем, що легко виявляються і самоописуються. Вибираючи архітектуру для веб-системи, важливо ретельно проаналізувати вимоги та обмеження програми, щоб визначити, чи підходить WOA найкраще.

Висновки до розділу

Функціональні вимоги - це ті, що описують конкретні можливості та функції, які система повинна надавати для задоволення потреб користувачів. Ці вимоги часто випливають з історій користувачів або сценаріїв використання і можуть бути використані для керівництва проектуванням і розробкою системи.

З іншого боку, нефункціональні вимоги описують характеристики та обмеження, яких повинна дотримуватися система, щоб відповідати функціональним вимогам. Вони можуть включати такі речі, як продуктивність, масштабованість, безпека та зручність використання, і часто визначаються в термінах метрик або еталонів, які можна використовувати для вимірювання та оцінки системи.

Клієнт-серверна архітектура - це загальний підхід до проектування розподілених систем, де система поділяється на два основні компоненти: клієнтський компонент, який взаємодіє з користувачем і запитує послуги у сервера, і серверний компонент, який надає ці послуги і керує ресурсами системи. Така архітектура може забезпечити такі переваги, як масштабованість, модульність і відмовостійкість, але також має деякі недоліки, такі як підвищена складність і накладні витрати.

Веб-орієнтована архітектура (WOA) - це особливий тип клієнт-серверної архітектури, який розроблений для використання переваг стандартних протоколів і технологій Всесвітньої павутини. Системи WOA зазвичай реалізуються за допомогою RESTful API, які надають простий, заснований на ресурсах інтерфейс, доступ до якого можна отримати за допомогою стандартних методів HTTP. WOA може забезпечити такі переваги, як інтероперабельність, масштабованість і модульність, і може використовуватися в поєднанні з іншими архітектурними стилями, такими як мікросервіси.

Архітектура мікросервісів - це ще один підхід до проектування розподілених систем, де система складається з невеликих, незалежно розгорнутих сервісів, які працюють разом для забезпечення загальної функціональності системи. Кожен мікросервіс може бути реалізований з використанням принципів WOA, що забезпечує високу масштабованість і модульний дизайн. Однак цей підхід може також створювати певні проблеми, такі як підвищена складність і потреба в ефективному виявленні сервісів і комунікації.

Таким чином, вибір правильної системної архітектури та вимог має вирішальне значення для успіху будь-якого програмного проекту. Ретельно зваживши функціональні і нефункціональні вимоги та обравши відповідний стиль архітектури, розробники можуть створювати системи, які можна масштабувати, модулювати і підтримувати, а також задовольняти потреби своїх користувачів.

РОЗДІЛ 3. ОПИС КОМПОНЕНТІВ СИСТЕМИ

3.1 Клієнтська частина

Клієнтська частина системи управління каталогом інтернет-магазину коміксів відповідає за надання інтерактивного та зручного інтерфейсу, який дозволяє клієнтам переглядати, шукати та керувати списками коміксів. Цей компонент створено з використанням JavaScript, універсальної мови програмування, яка широко використовується для веб-розробки.[17]

Клієнтський додаток представляє інтуїтивно зрозумілий і візуально привабливий користувальницький інтерфейс, який дозволяє клієнтам взаємодіяти з каталогом коміксів. Інтерфейс складається з різних компонентів, таких як навігаційні меню, пошукові рядки, списки коміксів і фільтри, призначені для покращення користувацького досвіду і полегшення навігації. Дизайн інтерфейсу розроблюється за допомогою CSS, в якому можна налаштувати весь зовнішній вигляд (див. додаток А).

Додаток відображає повний список доступних коміксів у візуально привабливій формі. Кожен список коміксів містить такі важливі деталі, як назва, автор, зображення обкладинки, видавець, дата виходу та ціна. Списки організовані та представлені у зручному для користувача форматі, що дозволяє покупцям швидко сканувати та оцінювати свої варіанти.

Клієнтський додаток надає потужні можливості пошуку та фільтрації, щоб допомогти користувачам знайти конкретні комікси відповідно до їхніх уподобань. Клієнти можуть шукати за назвою, автором, видавництвом, жанром або будь-якими іншими відповідними критеріями. Крім того, такі фільтри, як діапазон цін, дата випуску та популярність, дозволяють користувачам ще більше уточнити результати пошуку.

Коли покупець натискає на список коміксів, додаток представляє детальний перегляд обраного коміксу. Цей перегляд включає збільшене зображення обкладинки, анотацію, інформацію про автора, відомості про видання, відгуки покупців та відповідні рекомендації. Така комплексна

презентація дозволяє покупцям приймати обґрунтовані рішення щодо купівлі коміксів.

Клієнтський додаток легко інтегрується з функціоналом кошика для покупок, що дозволяє покупцям додавати обрані комікси до свого кошика для придбання. Кошик містить короткий опис обраних товарів разом із загальною вартістю, а також дозволяє клієнтам змінювати кількість або видаляти товари за потреби. Інтеграція з безпечними платіжними шлюзами забезпечує безперебійний і безпечний процес оформлення замовлення [19].

Щоб забезпечити персоналізований досвід, додаток включає в себе функції автентифікації користувачів та управління обліковими записами. Клієнти можуть створювати облікові записи, безпечно входити в систему та керувати своєю особистою інформацією, історією замовлень і списками бажань. Ця функціональність дозволяє користувачам відстежувати свої покупки, зберігати улюблені комікси та отримувати персоналізовані рекомендації.

Клієнтський додаток містить функції соціальної взаємодії, що сприяють залученню та створенню спільноти. Клієнти можуть оцінювати та переглядати комікси, залишати коментарі та ділитися улюбленими виданнями в соціальних мережах. Ця функціональність заохочує взаємодію між клієнтами і допомагає створити яскраву та активну спільноту любителів коміксів.

Клієнтський додаток розроблено адаптивно, що забезпечує оптимальну зручність використання та відображення на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та мобільні телефони. Інтерфейс адаптується до різних розмірів, роздільної здатності та орієнтації екрану, забезпечуючи послідовний і приємний досвід для клієнтів незалежно від пристрою, яким вони користуються [12].

Для підвищення загальної продуктивності та швидкості роботи додатку застосовуються різні методи оптимізації. До них відносяться лініве завантаження коміксів для покращення початкового часу завантаження, механізми кешування для мінімізації запитів до сервера та методи оптимізації

коду для зменшення часу завантаження та виконання сторінок. Ці оптимізації забезпечують плавний та ефективний перегляд сторінок для клієнтів.

Клієнтський додаток системи управління каталогом інтернет-магазину коміксів на основі JavaScript пропонує інтерактивний і зручний інтерфейс для клієнтів. Завдяки повному списку коміксів, розширеним можливостям пошуку та фільтрації, безпечній інтеграції з кошиком для покупок, автентифікації користувачів та функціям управління обліковими записами, додаток забезпечує безперебійний та приємний досвід для ентузіастів коміксів. Адаптивний дизайн забезпечує сумісність на різних пристроях, а оптимізація продуктивності

3.2 Серверна частина

За допомогою JavaScript створюється серверний додаток для розробки системи управління онлайн-каталогом коміксів. Ця програма призначена для підтримки інтернет-магазину коміксів, і вона буде побудована на системі користувача.

Код JavaScript написаний за допомогою фреймворку React (див. додаток Б).

Розберемо код та його компоненти:

- Імпорти:

React та useState імпортуються з бібліотеки React.

'./App.css', './data', та './data2' - це шляхи до інших файлів у проекті, які містять компоненти та дані.

- Компонент App:

Це головний компонент додатку, який буде відображати інтерфейс користувача та управляти станом додатку.

Він використовує хук useState, щоб оголосити декілька станів, таких як cartItems, activeTab, newProduct, products, packages та showProduct.

cartItems - масив елементів кошика.

activeTab - активна вкладка магазину.

`newProduct` - об'єкт, який містить дані про новий продукт, який додається до магазину.

`products` та `packages` - масиви продуктів коміксів та наборів.

`showProduct` - продукт, який показується з додатковою інформацією.

- Функції обробки подій:

`handleInputChange` - обробляє зміну значень у вхідних полях форми для додавання нового продукту. Вона оновлює стан `newProduct`.

`handleFormSubmit` - обробляє подання форми для додавання нового продукту. Вона створює новий об'єкт продукту зі значеннями з `newProduct` і додає його до `products` або `packages`, залежно від активної вкладки.

`addProductToCart` - додає продукт до кошика. Вона перевіряє, чи продукт вже присутній в кошику, і збільшує його кількість або додає новий елемент до кошика.

`removeProductFromCart` - видаляє продукт з кошика за його ідентифікатором.

`increaseProductQuantity` - збільшує кількість продукту в кошику на 1.

`decreaseProductQuantity` - зменшує кількість продукту в кошику на 1 або видаляє його, якщо кількість стає менше або дорівнює 0.

`getCartSubtotal`, `getCartTax`, `getCartTotal` - функції, які обчислюють підсумок, податок та загальну суму кошика.

`clearCart` - очищає кошик.

- Функції та методи відображення:

`handleProductHover` та `handleProductLeave` - обробляють події наведення курсору на продукт для відображення його додаткової інформації.

`renderProductInfo` - відображає додаткову інформацію про продукт, якщо він вибраний.

`renderProducts` - відображає групи продуктів у вигляді рядків.

`renderCartItem` - відображає елементи кошика.

`renderCart` - відображає вміст кошика разом із загальною сумою та кнопками.

- Повернення:

Рендериться розмітка додатку, включаючи заголовок, навігаційне меню, головну частину (яка залежить від активної вкладки) та футер.

У вкладці "Comics" відображаються продукти коміксів разом із формою для додавання нового продукту.

У вкладці "Set" відображаються продукти наборів разом із формою для додавання нового продукту.

У вкладці "Cart" відображається кошик з продуктами та загальною сумою.

Серверний додаток відіграє вирішальну роль в обробці запитів, обробці даних та управлінні каталогом коміксів. Він забезпечує безпечне і надійне середовище для взаємодії користувачів з інтернет-магазином і виконання різних операцій, таких як перегляд, пошук і купівля коміксів. Ось огляд основних компонентів і функціональних можливостей додатку:

Веб-сервер: Серверний додаток використовує фреймворк веб-сервера, такий як Node.js, для обробки вхідних HTTP-запитів і надання відповідей. Він прослуховує вказаний порт і перенаправляє запити до відповідних кінцевих точок.

Кінцеві точки API: Додаток надає набір кінцевих точок API, які дозволяють клієнтам (веб- або мобільним) взаємодіяти з сервером. Ці кінцеві точки полегшують такі операції, як отримання деталей коміксів, додавання товарів до кошика, обробка замовлень та управління обліковими записами користувачів.

Інтеграція з базами даних: Для постійного зберігання даних сервер інтегрується з системою управління базами даних, наприклад, MySQL, MongoDB або PostgreSQL. База даних зберігає інформацію про комікси, профілі користувачів, замовлення та інші відповідні дані. Серверна програма зв'язується з базою даних, щоб отримати та оновити цю інформацію за потреби [19].

Аутентифікація та авторизація: Аутентифікація користувача є життєво важливим аспектом безпеки додатку. Сервер використовує механізми аутентифікації, такі як JSON Web Tokens (JWT) або аутентифікація на основі сеансу, для перевірки особи користувача та авторизації доступу до захищених ресурсів. Він надійно зберігає облікові дані користувачів і керує сеансами користувачів, щоб забезпечити аутентифікований доступ.

Управління каталогом коміксів: Серверна програма надає функціональність для керування каталогом коміксів. Він дозволяє адміністраторам додавати, редагувати та видаляти записи коміксів. Ці операції змінюють записи бази даних, пов'язані з коміксами, забезпечуючи актуальність каталогу.

Пошук і фільтрація: Сервер дозволяє користувачам шукати і фільтрувати комікси за різними критеріями, такими як назва, автор, жанр або дата випуску. Він обробляє пошукові запити, витягує відповідні результати з бази даних і повертає їх клієнту для відображення.

Кошик і обробка замовлень: Сервер полегшує користувачам створення та управління кошиками для покупок. Він дозволяє користувачам додавати товари до кошика, змінювати кількість і переходити до оформлення замовлення. Сервер розраховує загальну ціну, застосовує знижки, якщо це можливо, і генерує замовлення для обробки.

Інтеграція платіжного шлюзу: Для обробки безпечних платіжних транзакцій сервер інтегрується з платіжним шлюзом, таким як PayPal, Stripe або Braintree. Він безпечно передає платіжні дані користувача платіжному шлюзу і отримує підтвердження або оновлення статусу після успішних або невдалих транзакцій [11].

Обробка помилок та ведення логів: Серверний додаток включає в себе надійні механізми обробки помилок, щоб ефективно обробляти винятки та помилки. Він реєструє відповідну інформацію, таку як повідомлення про помилки, трасування стеку та деталі запитів, щоб допомогти в усуненні несправностей та налагодженні.

Продуктивність і масштабованість: Сервер розроблений для оптимізації продуктивності та масштабованості. Він використовує технології кешування, такі як Redis, для зберігання даних, до яких часто звертаються, і мінімізації запитів до бази даних. Крім того, він може бути розгорнутий у розподіленому середовищі з використанням балансувальників навантаження та методів кластеризації для обробки збільшеного трафіку та забезпечення високої доступності.

Управління користувачами: Серверна програма включає в себе функцію управління користувачами. Вона дозволяє користувачам створювати облікові записи, оновлювати свої профілі та керувати своїми уподобаннями. Інформація про користувачів, така як імена користувачів, адреси електронної пошти та зашифровані паролі, надійно зберігається в базі даних.

Контроль доступу на основі ролей: Сервер реалізує контроль доступу на основі ролей (RBAC) для управління дозволами користувачів. Різні ролі, такі як адміністратори, клієнти та співробітники, мають різні рівні доступу до різних частин програми. RBAC гарантує, що лише авторизовані користувачі можуть виконувати певні дії, наприклад, керувати каталогом коміксів або отримувати доступ до конфіденційної інформації [12].

Завантаження зображень та файлів: Серверний додаток підтримує завантаження зображень і файлів для коміксів, обкладинок і аватарів користувачів. Він обробляє перевірку, зберігання та пошук цих файлів. Зображення можуть бути змінені або оптимізовані для забезпечення ефективного зберігання та відображення на стороні клієнта.

Аналітика та звітність: Сервер збирає та аналізує дані для створення звітів та аналітики про роботу інтернет-магазину. Він відстежує такі показники, як залученість користувачів, популярні комікси, тенденції продажів та рівень запасів. Ця аналітика допомагає адміністраторам приймати обґрунтовані рішення щодо маркетингу, управління запасами та бізнес-стратегій.

Сповіщення електронною поштою: Сервер може надсилати користувачам сповіщення на електронну пошту про різні події, такі як підтвердження

замовлень, оновлення доставки, скидання паролів або рекламні пропозиції. Інтеграція з поштовими сервісами, такими як SendGrid або SMTP-сервери, забезпечує надійну доставку транзакційних і маркетингових електронних листів.

Інтернаціоналізація та локалізація: Сервер підтримує інтернаціоналізацію та локалізацію для обслуговування користувачів з різних регіонів. Він може працювати з різними мовами, валютами та часовими поясами. Сервер вибирає відповідні переклади та формати на основі вподобань користувача або налаштувань браузера [12].

Заходи безпеки: Сервер включає в себе заходи безпеки для захисту від поширених веб-вразливостей. Він знезаражує дані, що вводяться користувачем, впроваджує заходи проти атак на міжсайтовий скриптинг (XSS) і SQL-ін'єкції, а також забезпечує безпечний зв'язок за протоколом HTTPS. Для усунення нових загроз і вразливостей проводяться регулярні аудити безпеки та оновлення.

Журналювання та аудит: Сервер реєструє важливі події, помилки та дії користувачів з метою аудиту та моніторингу. Він записує такі деталі, як запити API, дії користувачів і системні події. Ці журнали можна використовувати для усунення несправностей, аналізу продуктивності та досліджень безпеки.

Інтеграція із зовнішніми сервісами: Сервер може інтегруватися із зовнішніми сервісами, такими як платформи соціальних мереж або сторонні бази даних коміксів, щоб розширити функціональність. Наприклад, користувачі можуть мати можливість ділитися улюбленими коміксами в соціальних мережах або імпортувати свої списки читання з інших платформ.

Безперервна інтеграція та розгортання: Серверний додаток може бути інтегрований з конвеєром безперервної інтеграції/безперервного розгортання (CI/CD). Це дозволяє автоматизувати створення, тестування та розгортання оновлень для забезпечення безперебійного робочого процесу розробки та розгортання. Це допомагає підтримувати якість коду і полегшує часті оновлення та виправлення помилок.

Ці додаткові функції та можливості розширюють можливості серверного додатку для системи управління онлайн-каталогом коміксів. Вони сприяють комплексному та зручному користувацькому досвіду, забезпечуючи при цьому безпеку, масштабованість та ефективне управління операціями магазину коміксів.

Архітектура бази даних має на меті забезпечити надійну та масштабовану основу для системи управління онлайн-каталогом коміксів. Вона включає в себе визначення структури бази даних, вибір відповідної системи управління базами даних (СКБД) і проектування взаємозв'язків між різними сутностями. Ось огляд основних аспектів архітектури бази даних:

Система керування базами даних (СКБД): Вибір правильної СКБД має вирішальне значення. Найчастіше вибирають реляційні системи баз даних, такі як MySQL, PostgreSQL або Oracle, або NoSQL бази даних, такі як MongoDB або Cassandra. Вибір залежить від таких факторів, як структура даних, вимоги до масштабованості, потреби в продуктивності та обізнаність команди розробників.

Моделювання "сутність-зв'язок" (ER): ER-моделювання допомагає зрозуміти і визначити структуру бази даних. Воно передбачає визначення сутностей (таких як комікси, користувачі, замовлення) та їхніх атрибутів (назва, автор, ціна). Зв'язки між сутностями (один-до-одного, один-до-багатьох, багато-до-багатьох) встановлюються для забезпечення цілісності та узгодженості даних [11].

Таблиці та поля: Таблиці створюються для представлення сутностей, а поля (стовпці) визначають атрибути цих сутностей. Кожне поле має тип даних (наприклад, рядок, ціле число, дата) і обмеження (наприклад, первинний ключ, зовнішній ключ, унікальний) для забезпечення цілісності даних. Ретельна увага приділяється індексації полів для ефективного пошуку та сортування.

Зв'язки та об'єднання: Зв'язки між сутностями встановлюються за допомогою зовнішніх ключів. Об'єднання використовуються для ефективного отримання пов'язаних даних з декількох таблиць. Тип зв'язку (один-до-одного,

один-до-багатьох, багато-до-багатьох) визначає стратегію об'єднання (внутрішнє об'єднання, ліве об'єднання, праве об'єднання), яка використовується для об'єднання даних з різних таблиць.

Перевірка даних та обмеження: Правила та обмеження перевірки даних застосовуються для забезпечення цілісності даних і запобігання недійсним або неузгодженим даним. Ці обмеження можуть включати типи даних, перевірку діапазонів, обов'язкові поля, унікальність та обмеження посилальної цілісності.

Оптимізація продуктивності: Для підвищення продуктивності бази даних використовуються такі методи, як індексування, оптимізація запитів і кешування. Індеси створюються на часто використовуваних полях для прискорення пошуку даних. Оптимізація запитів включає в себе аналіз і тонку настройку SQL-запитів для ефективного виконання. Методи кешування можуть використовуватися для зберігання даних, до яких часто звертаються, в пам'яті для швидшого пошуку.

Резервне копіювання та відновлення: Регулярне резервне копіювання баз даних необхідне для захисту від втрати даних. Стратегії резервного копіювання включають повне резервне копіювання, інкрементне резервне копіювання або створення знімків. Плани відновлення розробляються на випадок пошкодження даних або системних збоїв, гарантуючи, що база даних може бути відновлена до узгодженого стану.

Заходи безпеки: Заходи безпеки застосовуються для захисту бази даних від несанкціонованого доступу та витоку даних. Вони включають автентифікацію користувачів, контроль доступу на основі ролей (RBAC), шифрування конфіденційних даних і механізми аудиту для моніторингу та відстеження дій користувачів.

Масштабованість і реплікація: Архітектура бази даних повинна бути розроблена з урахуванням можливості масштабування. Такі методи, як реплікація бази даних, шардінг або кластеризація, використовуються для розподілу робочого навантаження на базу даних і обробки зростаючих обсягів даних і трафіку користувачів.

Архівування та очищення даних: Зі зростанням каталогу коміксів важливо впроваджувати стратегії архівування та очищення даних. Історичні дані або дані, до яких рідше звертаються, можна архівувати в окреме сховище або перемістити до системи зберігання нижчого рівня, щоб оптимізувати продуктивність і зменшити витрати. Механізми очищення даних впроваджуються для видалення застарілих або неактуальних даних з бази даних.

Моніторинг та оптимізація продуктивності: Архітектура бази даних включає в себе інструменти та методи моніторингу для відстеження продуктивності, працездатності та доступності системи баз даних. Показники продуктивності, такі як час відгуку, пропускна здатність запитів і використання ресурсів, відстежуються для виявлення вузьких місць і оптимізації продуктивності. Налагоджено механізми оповіщення, які сповіщають адміністраторів про будь-які критичні проблеми або аномалії.

Аварійне відновлення та висока доступність: Для забезпечення безперервності бізнесу впроваджуються стратегії аварійного відновлення. Це передбачає регулярне резервне копіювання, віддалене зберігання резервних копій та можливість відновлення бази даних до попереднього стану в разі втрати даних або збоїв системи. Конфігурації високої доступності, такі як реплікація бази даних або відмовостійкі кластери, використовуються для мінімізації часу простою та забезпечення безперервного доступу до бази даних.

Документація та словник даних: Належне документування архітектури бази даних має важливе значення для подальшого обслуговування, оновлення та усунення несправностей. Створюється словник даних, який описує структуру та атрибути кожної таблиці та їх взаємозв'язки. Ця документація слугує довідником для розробників, адміністраторів та інших зацікавлених сторін.

Відповідність та конфіденційність даних: Архітектура бази даних враховує положення про конфіденційність даних та вимоги до комплаєнсу. Вживаються заходи для забезпечення безпеки та захисту даних користувачів, такі як методи анонімізації, шифрування даних та контроль доступу.

Розглядається і вирішується питання відповідності нормативним актам, таким як GDPR, CCPA або HIPAA.

Інтеграція з прикладними рівнями: Архітектура бази даних повинна безперешкодно інтегруватися з серверним додатком та іншими рівнями додатків, такими як інтерфейс користувача та зовнішні сервіси. API або рівні доступу до даних призначені для полегшення зв'язку між додатком і базою даних, забезпечуючи ефективний пошук і модифікацію даних.

Масштабованість і розширюваність у майбутньому: Архітектура бази даних розроблена з урахуванням майбутнього зростання, змін та нових функцій. Гнучкість вбудована в структуру для підтримки додавання нових сутностей, атрибутів або зв'язків. Належні стратегії індексування, розбиття на розділи або кешування реалізуються для обробки зростаючих обсягів даних і паралельної роботи користувачів.

Таким чином, проектування архітектури бази даних передбачає ретельне планування її структури, взаємозв'язків і методів оптимізації продуктивності. Такі міркування, як вибір системи управління базами даних, нормалізація, індексування, заходи безпеки, масштабованість і стратегії міграції даних, сприяють створенню надійної та ефективної архітектури бази даних. Ця архітектура формує основу для системи управління онлайн-каталогом коміксів, забезпечуючи цілісність даних, продуктивність і масштабованість.

Для своєї роботи я обрав СКБД MySQL. ER-модель має наступний вигляд (рис 3.1).

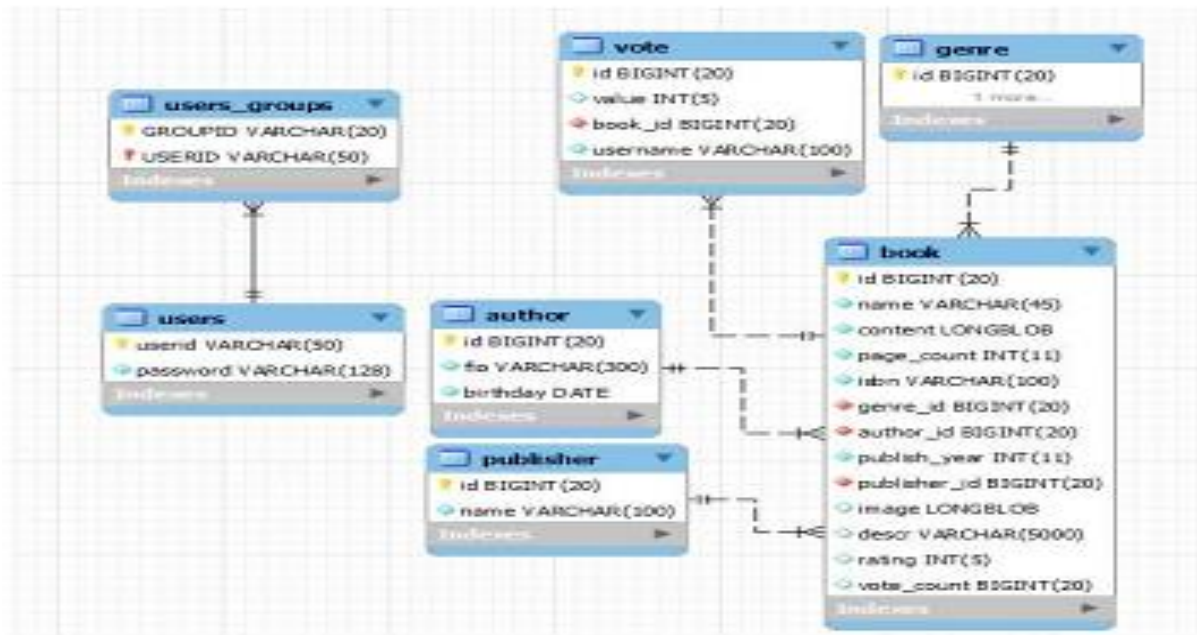


Рис. 3.1. Модель спроектованої бази даних

Модель бази даних складається з наступних таблиць: book, genre, vote, author, publisher, users, users_groups. Опис кожної таблиці подано нижче (див. табл 3.1).

Таблиця 3.1

Опис таблиць бази даних

№ пп	Назва таблиці	Призначення
1	BOOK	Загальна інформація про комікс
2	GENRE	Перелік жанрів
3	VOTE	Перелік голосування
4	AUTHOR	Перелік авторів
5	PUBLISHER	Перелік видавництв
6	USERS	Перелік паролів користувачів

7	USERS_GROUP	Перелік користувачів
---	-------------	----------------------

Таблиця BOOK є найбільш інформаційно-змістовною, вона містить основну інформацію про комікс: первинний ключ, назву, опис, кількість сторінок і т.д. (див. табл. 3.2).

Таблиця 3.2

Опис таблиці BOOK

№пп	Назва поля	Тип	Призначення
1	id	bigint(20)	Первинний ключ
2	name	varchar(45)	Назва коміксу
3	content	longblob	Основний контент
4	page_count	int(11)	Кількість сторінок
5	genre_id	bigint(20)	Ідентифікатор жанру
6	author_id	bigint(20)	Ідентифікатор автору
7	publish_year	int(11)	Рік видання
8	publisher_id	bigint(20)	Ідентифікатор видавництва
9	image	longblob	Обкладинка
10	rating	int(5)	Рейтинг коміксу
11	vote_count	bigint(20)	Кількість голосів

Таблиця GENRE має інформацію про назви жанрів (див. Рис. 3.3).

Таблиця 3.3

Опис таблиці GENRE

№пп	Назва поля	Тип	Призначення
1	id	bigint(20)	Первинний ключ

2	name	varchar(45)	Назва жанру
---	------	-------------	-------------

Таблиця VOTE має інформацію про голосування користувачів та їх оцінки коміксів (див. рис. 3.4).

Таблиця 3.4

Опис таблиці VOTE

№пп	Назва поля	Тип	Призначення
1	id	bigint(20)	Первинний ключ
2	value	int(5)	Оцінка
3	book_id	bigint(20)	Ідентифікатор коміксу
4	username	varchar	Ім'я користувача

Таблиця AUTHOR має інформацію про автора коміксу та його дату народження (див. рис. 3.5).

Таблиця 3.5

Опис таблиці AUTHOR

№пп	Назва поля	Тип	Призначення
1	id	bigint(20)	Первинний ключ
2	name	varchar	Ім'я автора коміксу
3	birthday	date	День народження

Таблиця PUBLISHER має інформацію про назву видавництва (див. рис. 3.6).

Таблиця 3.6

Опис таблиці PUBLISHER

№пп	Назва поля	Тип	Призначення
1	id	bigint(20)	Первинний ключ

2	name	varchar	Назва видавництва
---	------	---------	-------------------

Таблиця USERS має інформацію про паролі користувачів (див. рис. 3.7).

Таблиця 3.7

Опис таблиці USERS

№пп	Назва поля	Тип	Призначення
1	userid	bigint(20)	Первинний ключ
2	password	varchar	Пароль

Таблиця USERS_GROUPS має інформацію про користувачів (див. рис. 3.8).

Таблиця 3.8

Опис таблиці USERS_GROUPS

№пп	Назва поля	Тип	Призначення
1	groupid	bigint(20)	Первинний ключ
2	userid	varchar	Ідентифікатор користувача

3.3. Процес додавання нового коміксу в каталог

Збір інформації про комікс: Процес починається зі збору всієї необхідної інформації про новий комікс. Сюди входять такі дані, як назва коміксу, автор, ілюстратор, дата публікації, жанр, синопсис, зображення обкладинки та будь-які додаткові метадані.

Перевірка та введення даних: Зібрана інформація перевіряється для забезпечення її точності та повноти. Серверна програма виконує перевірки, щоб переконатися, що необхідні поля заповнені, формати даних правильні, а всі обмеження дотримані. Після перевірки дані вносяться до бази даних.

Завантаження зображення обкладинки: Зображення обкладинки коміксу завантажуються на сервер. Серверна програма обробляє завантаження

зображення, гарантуючи, що воно відповідає необхідним специфікаціям, таким як формат файлу (наприклад, JPEG, PNG), обмеження за розміром та роздільна здатність. Зображення зберігається у визначеному місці на сервері або в хмарному сховищі.

Присвоєння ідентифікатора коміксу: Новому коміксу присвоюється унікальний ідентифікатор (ID). Цей ідентифікатор слугує первинним ключем у базі даних, що дозволяє легко посилатися на нього та знаходити інформацію про комікс.

Вставка до бази даних: Перевірена інформація про комікс разом з присвоєним ідентифікатором вставляється в базу даних. Серверна програма виконує відповідні запити до бази даних, щоб додати новий запис до таблиці коміксів. Атрибути коміксу, такі як назва, автор, дата публікації, жанр і URL-адреса обкладинки, зберігаються у відповідних полях.

Оновлення зв'язків: Якщо новий комікс має зв'язки з іншими сутностями, такими як автори або жанри, відповідні зв'язки встановлюються або оновлюються в базі даних. Наприклад, якщо автор є новим, його інформація може бути додана до таблиці авторів і пов'язана з коміксом за допомогою зовнішнього ключа.

Індексування та можливість пошуку: Для забезпечення ефективного пошуку та фільтрації коміксів створюються відповідні індекси за такими полями, як назва, автор та жанр. Ці індекси оптимізують виконання запитів, дозволяючи швидше знаходити дані на основі конкретних критеріїв пошуку.

Оновлення каталогу та анулювання кешу: Після успішного додавання коміксу до бази даних каталог оновлюється, щоб відобразити нове додавання. Серверна програма може регенерувати каталог або оновити певні розділи, щоб відобразити щойно доданий комікс. Якщо реалізовано механізми кешування, кеш анулюється, щоб користувачі бачили останню версію каталогу.

Інтеграція з інтерфейсом: Серверна програма взаємодіє з інтерфейсом користувача, щоб повідомити йому про новий комікс. Це може включати

надсилання повідомлень або запуск оновлень, щоб користувачі могли побачити новий комікс у каталозі.

Забезпечення якості та тестування: Перед тим, як зробити новий комікс видимим для користувачів, проводиться ретельне тестування і забезпечення якості. Це включає перевірку правильності відображення коміксу в каталозі, перевірку пошукової функціональності та забезпечення належної роботи всіх супутніх функцій і взаємодій.

Випуск і доступність: Після того, як новий комікс пройшов тестування і перевірку якості, він стає доступним для користувачів. Він з'являється в каталозі, і користувачі можуть шукати, переглядати і, можливо, купувати або додавати його до своїх колекцій.

Процес додавання нового коміксу до каталогу включає збір і перевірку інформації про комікс, завантаження зображення обкладинки, присвоєння унікального ідентифікатора, внесення даних до бази даних, оновлення зв'язків, індексування для пошуку, оновлення каталогу, інтеграцію з інтерфейсом, тестування і надання коміксу користувачам. Кожен крок забезпечує точне представлення та плавну інтеграцію нового коміксу в систему управління онлайн-каталогом коміксів.

3.4. Організація пошуку коміксу в каталозі

Процес пошуку коміксів у каталозі передбачає надання користувачам можливості знайти конкретні комікси на основі їхніх уподобань та критеріїв пошуку.

Введення даних користувачем: Процес пошуку починається, коли користувач вводить свої критерії пошуку в інтерфейс пошуку, наданий серверною програмою. Користувач може ввести ключові слова, такі як назва коміксу, ім'я автора, жанр або будь-яку іншу релевантну інформацію, щоб звузити пошук.

Формування запиту: Серверна програма отримує пошуковий запит від користувача і генерує запит до бази даних на основі заданих критеріїв. Запит

призначений для вилучення з бази даних коміксів, які відповідають критеріям пошуку.

Пошук у базі даних: Серверна програма виконує сформований запит до бази даних каталогу коміксів. Пошуковий запит може включати пошук збігів у певних полях, таких як назва коміксу, ім'я автора або жанр, з використанням відповідних пошукових операторів (наприклад, точний збіг, частковий збіг, без урахування регістру).

Отримання результатів пошуку: Додаток на стороні сервера отримує результати пошуку, отримані в результаті запиту до бази даних. Результати пошуку включають комікси, які відповідають критеріям пошуку користувача, разом з відповідною інформацією, такою як назва, автор, дата публікації та URL-адреса зображення обкладинки.

Пагінація та обмеження: Якщо є велика кількість результатів пошуку, можна застосувати техніку пагінації, щоб представити результати у вигляді менших, зручних для керування фрагментів. Серверна програма отримує підмножину результатів пошуку на основі потрібного номера сторінки і кількості результатів, які потрібно відобразити на сторінці.

Фільтрація та сортування: Серверний додаток може надавати користувачам можливість фільтрувати та сортувати результати пошуку. Користувачі можуть уточнити свій пошук, застосовуючи додаткові фільтри на основі таких критеріїв, як жанр, діапазон дат публікації, автор або інші атрибути. Опції сортування дозволяють користувачам впорядкувати результати пошуку за релевантністю, датою публікації, назвою або будь-якими іншими критеріями.

Відображення результатів пошуку: Серверна програма представляє результати пошуку користувачеві за допомогою інтерфейсу користувача. Результати пошуку відображаються у зручному для користувача вигляді, часто у вигляді списку або сітки, що містить релевантну інформацію, таку як назва коміксу, зображення обкладинки, автор та інші метадані.

Взаємодія та навігація: Користувач може взаємодіяти з результатами пошуку, щоб переглянути більше деталей про конкретний комікс. Натискання на результат пошуку може призвести до переходу на спеціальну сторінку коміксу, що містить вичерпну інформацію про комікс, включаючи синопсис, ціну, рецензії та додаткові зображення.

Уточнення пошуку: Серверна програма дозволяє користувачам уточнювати критерії пошуку, якщо початковий пошук не дав задовільних результатів. Користувачі можуть змінити пошуковий запит, застосувати додаткові фільтри або налаштувати параметри сортування, щоб ще більше звужити пошук і знайти потрібний комікс.

Пошукові пропозиції: Коли користувачі вводять свої пошукові запити, додаток на стороні сервера може надавати пошукові підказки або функцію автозавершення, щоб допомогти їм. Ці пропозиції генеруються на основі популярних пошукових термінів, нещодавно введених ключових слів або інших релевантних даних, щоб покращити користувацький досвід і допомогти користувачам легше знаходити відповідні комікси.

Оптимізація продуктивності: Серверний додаток реалізує методи оптимізації продуктивності для забезпечення ефективних пошукових операцій. Це може включати індексування відповідних полів, кешування результатів пошуку або використання технологій пошукової системи для забезпечення швидших і точніших результатів пошуку.

Постійне вдосконалення: Серверна програма збирає дані про пошукові шаблони і поведінку користувачів, що дозволяє постійно вдосконалювати пошукову функціональність. Дані про взаємодію з користувачами та їхні відгуки можна аналізувати для вдосконалення алгоритму пошуку, уточнення рейтингу релевантності та забезпечення більш персоналізованого й ефективного пошуку.

3.5 Процес оформлення замовлення на комікс

Процес оформлення замовлення на комікс полягає у спрощенні вибору, кастомізації та купівлі коміксу в інтернет-магазині.

Вибір коміксу: Користувач переглядає каталог або здійснює пошук, щоб знайти потрібний комікс. Він може переглянути деталі, зображення обкладинки, ціну та іншу важливу інформацію про комікс. Після того, як користувач визначився з конкретним коміксом, він переходить до процесу оформлення замовлення.

Додавання до кошика: користувач додає обраний комікс до свого кошика. Додаток на стороні сервера обробляє цю дію, зберігаючи необхідну інформацію про комікс, таку як ідентифікатор коміксу, кількість, ціну та будь-які параметри налаштування, в сеансі користувача або базі даних.

Перегляд кошика: Користувач переглядає вміст свого кошика, який відображає короткий опис вибраних коміксів, кількість і загальну ціну. Користувач може змінити кількість, видалити товари або перейти до оформлення замовлення.

Інформація про покупця: Користувач надає інформацію про себе, необхідну для оформлення замовлення, таку як ім'я, адреса доставки, адреса електронної пошти та контактні дані. Ця інформація безпечно збирається додатком на стороні сервера для обробки замовлення і зв'язку з користувачем щодо покупки.

Вибір способу доставки: Користувач обирає бажаний спосіб доставки з доступних варіантів, запропонованих інтернет-магазином. Це можуть бути різні транспортні компанії, такі як стандартна доставка, експрес-доставка або місцевий самовивіз, а також пов'язані з ними витрати та орієнтовний час доставки.

Вибір способу оплати: Користувач обирає спосіб оплати, який він бажає використовувати для замовлення. Серверний додаток інтегрується з різними платіжними шлюзами, пропонуючи такі варіанти, як оплата кредитною картою, PayPal або інші електронні способи оплати. Користувач може також

мати можливість застосувати коди знижок або подарункові картки, якщо вони доступні.

Огляд замовлення: Користувачеві надається коротка інформація про його замовлення, включаючи вибрані комікси, кількість, загальну ціну, інформацію про клієнта, спосіб доставки та спосіб оплати. Користувач може переглянути всі деталі перед тим, як підтвердити замовлення.

Підтвердження замовлення: Після того, як користувач підтверджує замовлення, додаток на стороні сервера обробляє замовлення, генеруючи унікальний ідентифікатор замовлення і записуючи деталі замовлення в базу даних. Користувач отримує електронний лист з підтвердженням замовлення або повідомлення, що містить інформацію про замовлення, платіжні реквізити та орієнтовний термін доставки.

Обробка платежів: Якщо обраний спосіб оплати вимагає авторизації в режимі реального часу, серверний додаток зв'язується з платіжним шлюзом для безпечної обробки платіжної транзакції. Користувача може бути перенаправлено на захищену сторінку оплати або запропоновано ввести платіжні реквізити безпосередньо на сторінці оформлення замовлення в інтернет-магазині.

Виконання замовлення: Після успішної обробки платежу додаток на стороні сервера ініціює процес виконання замовлення. Це включає в себе пакування коміксу, створення транспортної етикетки та координацію з обраним перевізником для доставки. Користувач може отримувати сповіщення про доставку або інформацію про відстеження, щоб відстежувати прогрес свого замовлення.

Історія замовлень та відстеження: Додаток на стороні сервера зберігає історію замовлень користувача, дозволяючи йому переглядати і відстежувати статус своїх замовлень. Користувач може отримати доступ до своєї історії замовлень через свій обліковий запис або отримувати оновлення електронною поштою чи сповіщеннями.

Підтримка та комунікація з клієнтами: Протягом усього процесу оформлення замовлення додаток на стороні сервера надає можливості для підтримки та спілкування з клієнтом. Це може включати контактну інформацію, поширені запитання, чат або систему квитків підтримки для вирішення будь-яких запитань або проблем, пов'язаних із замовленням.

Процес повернення та відшкодування: Якщо користувач бажає повернути або обміняти комікс, серверний додаток полегшує процес повернення та відшкодування. Політика та процедури повернення повідомляються користувачеві, а додаток виконує необхідні дії, щоб ініціювати процес повернення, відшкодування або обміну.

Завершення замовлення та зворотній зв'язок: Коли користувач отримує замовлений комікс, він може позначити замовлення як завершене через інтерфейс інтернет-магазину. Додаток на стороні сервера може запропонувати користувачеві залишити відгук або рецензію про придбаний комікс і загальний досвід замовлення.

Управління запасами: Після оформлення замовлення серверний додаток оновлює систему інвентаризації, щоб відобразити зменшену кількість замовлених коміксів. Це забезпечує точне управління запасами та запобігає перепродажу.

Аналітика та звітність: Серверний додаток збирає дані про метрики замовлень, такі як загальний обсяг продажів, популярні комікси та демографічні дані клієнтів. Ці дані можна використовувати для створення звітів, аналізу тенденцій та прийняття обґрунтованих бізнес-рішень.

Відстеження замовлень та сповіщення клієнтів: Додаток на стороні сервера надає функцію відстеження замовлень, що дозволяє користувачам відстежувати прогрес їхнього відправлення. Він надсилає сповіщення клієнтам на різних етапах процесу виконання замовлення, таких як підтвердження замовлення, підтвердження доставки та оновлення доставки.

Підтримка замовлень та обробка повернень: Якщо у користувачів виникають проблеми із замовленням або їм потрібна допомога, серверний

додаток надає канали підтримки клієнтів. Агенти підтримки можуть отримати доступ до деталей замовлення і допомогти клієнтам у вирішенні питань, пов'язаних з доставкою, пошкодженими товарами або розбіжностями в замовленні.

Безпека та конфіденційність: Серверний додаток забезпечує безпеку та конфіденційність інформації про клієнта протягом усього процесу оформлення замовлення. Він використовує методи шифрування та дотримується правил захисту даних, щоб захистити конфіденційні дані, такі як платіжні реквізити та особисту інформацію.

Постійне вдосконалення: Додаток на стороні сервера збирає відгуки та рецензії від клієнтів, щоб покращити процес замовлення. Інформація, отримана з відгуків клієнтів, може бути використана для покращення користувацького досвіду, спрощення процесу оформлення замовлення та усунення будь-яких проблемних моментів.

Інтеграція із зовнішніми системами: Серверний додаток інтегрується із зовнішніми системами, такими як транспортні компанії, платіжні шлюзи та системи управління запасами. Ця інтеграція забезпечує безперебійний зв'язок та обмін даними між інтернет-магазином та цими зовнішніми системами.

Масштабованість і продуктивність: Серверний додаток розрахований на обробку великого обсягу замовлень, забезпечуючи масштабованість і оптимальну продуктивність навіть у пікові періоди. Балансування навантаження, механізми кешування та оптимізація бази даних реалізовані для підтримки великої кількості одночасних користувачів і підтримки швидкості реагування.

Таким чином, процес оформлення замовлення на комікс включає в себе вибір бажаного коміксу, додавання його до кошика, надання інформації про покупця та оплату, перегляд замовлення, підтвердження покупки, обробку оплати, виконання замовлення та надання підтримки після покупки. Додаток на стороні сервера керує цими кроками, щоб забезпечити безперебійний і безпечний процес замовлення для користувачів інтернет-магазину коміксів.

Висновки до розділу

Загалом, ці процеси підкреслюють функціональність і потік додатку для управління каталогом коміксів. Серверний додаток разом із базою даних підтримує такі завдання, як додавання нових коміксів, пошук у каталозі та обробка замовлень. Ефективно реалізуючи ці процеси, додаток може забезпечити користувачам безперебійний і приємний досвід під час вивчення та купівлі коміксів.

РОЗДІЛ 4. РОЗГОРТАННЯ ТА НАЛАШТУВАННЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для успішного розгортання та налаштування системи управління каталогом коміксів необхідно враховувати певні вимоги до апаратного та програмного забезпечення.

Серверна інфраструктура: Для розміщення серверного додатку та пов'язаних з ним компонентів необхідні достатні апаратні ресурси. Це включає в себе надійну серверну машину або хмарну інфраструктуру, здатну впоратися з очікуваним робочим навантаженням і трафіком.

Обчислювальна потужність: серверне обладнання повинно мати достатню обчислювальну потужність для обробки одночасних запитів і ефективного виконання складних операцій. Процесор повинен відповідати вимогам до продуктивності системи для забезпечення безперебійної роботи. Необхідно мінімум 1-ядерний процесор з тактовою частотою не менше 1.6 ГГц. Рекомендовано більш потужний процесор для покращення продуктивності та обробки більшого навантаження.

Пам'ять (оперативна пам'ять): Достатній обсяг оперативної пам'яті необхідний для розміщення серверного додатку, бази даних та інших програмних компонентів. Обсяг оперативної пам'яті слід визначати, виходячи з очікуваної одночасної активності користувачів і розміру каталогу коміксів. Мінімум 2 ГБ оперативної пам'яті. Рекомендовано 4 ГБ або більше для забезпечення плавної роботи додатку та підтримки більшої кількості користувачів.

Місце для зберігання даних: Для зберігання каталогу коміксів, включаючи зображення, метадані, дані користувачів і будь-яку іншу важливу інформацію, потрібен значний обсяг пам'яті. Система зберігання повинна забезпечувати швидкий і надійний доступ до даних. Мінімум 10 ГБ вільного простору на жорсткому диску для збереження програмного забезпечення та даних.

Мережева інфраструктура: Стабільне та високошвидкісне мережеве з'єднання необхідне для безперебійного зв'язку між додатком на стороні сервера, базою даних та будь-якими зовнішніми системами, такими як платіжні шлюзи або служби доставки. Достатня пропускна здатність має вирішальне значення, особливо в періоди пікового навантаження.

Операційна система: Додаток на стороні сервера повинен бути сумісним з обраною операційною системою. Найпоширеніші варіанти включають Windows Server, дистрибутиви Linux (наприклад, Ubuntu, CentOS) або хмарні платформи, такі як Amazon Web Services (AWS) або Microsoft Azure.

Веб-сервер: Надійне програмне забезпечення веб-сервера, таке як Apache HTTP Server або Nginx, необхідне для обробки вхідних HTTP-запитів, перенаправлення їх до серверного додатку та надання відповідних відповідей.

Мова програмування та фреймворк: Серверний додаток розробляється з використанням JavaScript і може використовувати популярні фреймворки, такі як Node.js або Express.js. Обраний фреймворк повинен відповідати вимогам розробки та надавати необхідні функції для створення додатку.

Реляційна система управління базами даних (СКБД): Система потребує СКБД, наприклад, MySQL, PostgreSQL або Microsoft SQL Server, для зберігання та управління даними каталогу коміксів. Обрана система баз даних повинна бути масштабованою, надійною та підтримувати ефективний пошук даних і запити.

Бібліотеки та залежності: У серверному додатку можуть використовуватися різні бібліотеки та залежності JavaScript для виконання таких завдань, як автентифікація, перевірка даних, обробка зображень або інтеграція з API. Ці бібліотеки повинні бути сумісними з обраною мовою програмування та фреймворками.

Заходи безпеки: Система повинна включати заходи безпеки, включаючи протоколи шифрування (наприклад, HTTPS), безпечні методи кодування та контроль доступу для захисту даних користувачів, запобігання

несанкціонованому доступу та захисту конфіденційної інформації, наприклад, платіжних реквізитів.

Інструменти розробки: Інструменти розробки програмного забезпечення, включаючи інтегровані середовища розробки (IDE), редактори коду, системи контролю версій (наприклад, Git) та інструменти управління проектами, є важливими для ефективної розробки, налагодження, співпраці та обслуговування серверного додатку.

Моніторинг та аналітика: Інструменти для моніторингу продуктивності сервера, відстеження метрик використання та аналізу журналів додатків корисні для проактивного обслуговування, оптимізації продуктивності та усунення несправностей.

Сумісність і масштабованість: Серверний додаток і пов'язане з ним програмне забезпечення повинні бути сумісними з різними веб-браузерами і пристроями, щоб забезпечити безперебійну роботу користувачів. Система також повинна бути спроектована таким чином, щоб впоратися зі збільшенням трафіку і пристосуватися до майбутнього зростання, враховуючи такі опції, як балансування навантаження і горизонтальне масштабування.

Важливо враховувати ці вимоги до апаратного та програмного забезпечення під час розгортання та конфігурації, щоб забезпечити стабільність, продуктивність та безпеку системи. Дотримання цих вимог сприятиме створенню міцної та надійної системи управління каталогом коміксів.

4.2 Розгортання системи

Розгортання системи управління каталогом коміксів передбачає процес встановлення, налаштування та запуску системи на обраній серверній інфраструктурі.

Налаштування сервера: Почніть з налаштування серверної інфраструктури, будь то фізичний сервер або хмарна платформа. Встановіть

операційну систему, яка відповідає вимогам до програмного забезпечення. Налаштуйте мережеві параметри та переконайтеся, що сервер має надійне з'єднання з Інтернетом.

Встановлення програмного забезпечення: Встановіть необхідні програмні компоненти, такі як веб-сервер (наприклад, Apache або Nginx), середовище виконання мови програмування (наприклад, Node.js) та обрану систему управління базами даних (наприклад, MySQL або PostgreSQL). Дотримуйтесь інструкцій з встановлення, наданих кожним постачальником програмного забезпечення.

Конфігурація: Налаштуйте веб-сервер для роботи з додатком на стороні сервера. Сюди входить вказівка IP-адреси, порту та доменного імені сервера (якщо це можливо). Налаштуйте віртуальні хости або серверні блоки для обробки вхідних запитів і перенаправлення їх до відповідної програми.

Налаштування бази даних: Налаштуйте систему управління базою даних, створивши нову базу даних для каталогу коміксів. Налаштуйте параметри підключення до бази даних, включаючи назву бази даних, ім'я користувача та пароль. Імпортуйте необхідну схему бази даних або запустіть міграцію бази даних, щоб створити необхідні таблиці та зв'язки.

Розгортання програми: Скопіюйте код програми на стороні сервера на сервер. Переконайтеся, що необхідні залежності та бібліотеки встановлені та оновлені. Налаштуйте всі змінні оточення або конфігураційні файли, необхідні для роботи програми, наприклад, деталі підключення до бази даних або ключі API для зовнішніх сервісів.

Тестування: Проведіть ретельне тестування розгорнутої системи, щоб переконатися, що всі компоненти функціонують належним чином. Протестуйте різні функції, такі як додавання коміксів, пошук у каталозі та оформлення замовлень. Переконайтеся, що система поводить себе так, як очікувалося, і коректно обробляє помилки. Виправте будь-які помилки або проблеми, виявлені під час тестування.

Посилення безпеки: Впровадьте заходи безпеки для захисту системи. Це включає в себе увімкнення брандмауерів, налаштування SSL-сертифікатів для безпечного зв'язку (HTTPS), налаштування контролю доступу, а також застосування виправлень та оновлень безпеки на сервері та програмних компонентах. Регулярно контролюйте та оновлюйте систему для усунення вразливостей безпеки, що з'являються.

Масштабованість та оптимізація продуктивності: Розгляньте можливість впровадження заходів масштабованості, щоб впоратися зі збільшенням трафіку і пристосуватися до майбутнього зростання. Це може включати балансування навантаження для розподілу запитів між декількома серверами, механізми кешування для підвищення продуктивності та методи оптимізації бази даних, такі як індексування або оптимізація запитів.

Резервне копіювання та відновлення: Налаштуйте регулярне резервне копіювання системи для захисту від втрати даних. Впровадьте автоматизовані процеси резервного копіювання, які надійно зберігають резервні копії як локально, так і за межами сайту. Протестуйте процедури резервного копіювання та відновлення, щоб переконатися, що дані можуть бути відновлені в разі збою або катастрофи.

Документація та навчання: Задokumentуйте процес розгортання, включаючи конфігурації, залежності та будь-які налаштування, зроблені під час розгортання. Ця документація слугуватиме довідником для подальшого обслуговування та усунення несправностей. Проведіть навчання або надайте документацію для адміністраторів або працівників служби підтримки, які будуть керувати системою.

Моніторинг та обслуговування: Налаштуйте інструменти моніторингу для відстеження продуктивності системи, часу безвідмовної роботи та використання ресурсів. Слідкуйте за журналами сервера, продуктивністю бази даних та метриками конкретних додатків. Регулярно встановлюйте оновлення та патчі безпеки для програмних компонентів. Виконуйте рутинні завдання з обслуговування, такі як оптимізація бази даних та ротація лог-файлів.

Запуск і тестування після розгортання: Після того, як система повністю налаштована і протестована, вона готова до запуску для публічного використання. Уважно стежте за роботою системи протягом початкового періоду, щоб вирішити будь-які проблеми з продуктивністю або зручністю використання, які можуть виникнути. Збирайте відгуки від користувачів і вносьте необхідні корективи на основі їхнього досвіду.

Дотримуючись цих кроків розгортання, система управління каталогом коміксів може бути успішно розгорнута і доступна для користувачів. Регулярне обслуговування і моніторинг гарантують, що система залишатиметься стабільною, безпечною і оптимально працюватиме протягом тривалого часу.

4.3 Налаштування системи

Конфігурація системи передбачає встановлення різних параметрів і налаштувань в системі управління каталогом коміксів, щоб забезпечити її роботу відповідно до конкретних вимог.

Адміністративні налаштування: Налаштуйте адміністративні параметри, такі як контроль доступу до системи, ролі користувачів і дозволи. Визначте ролі для адміністраторів, менеджерів та інших типів користувачів з відповідними рівнями доступу до різних функціональних можливостей системи. Вкажіть політику паролів і заходи безпеки для облікових записів користувачів.

Конфігурація каталогу: Налаштуйте параметри каталогу коміксів, включно з макетом, зовнішнім виглядом та організацією записів коміксів. Налаштуйте категорії, жанри і теги, щоб допомогти користувачам ефективніше орієнтуватися і шукати комікси. Налаштуйте відображення деталей коміксу, включаючи зображення обкладинки, опис, інформацію про автора та ціну.

Налаштування цін і платежів: Налаштуйте структуру цін на комікси, включно зі звичайними цінами, знижками та акційними пропозиціями. Налаштуйте інтеграцію з платіжним шлюзом, щоб уможливити безпечні

онлайн-платежі. Вкажіть прийнятні способи оплати, такі як кредитні картки, PayPal або інші платіжні системи. Налаштуйте податкові параметри відповідно до чинного податкового законодавства.

Конфігурація доставки та доставки: Налаштуйте варіанти доставки, тарифи та способи для фізичних замовлень коміксів. Вкажіть зони доставки, перевізників та пов'язані з ними витрати. Встановіть правила міжнародної доставки, тарифи на обробку та відстеження замовлень. Інтегруйтеся з постачальниками послуг доставки, щоб автоматизувати створення транспортних етикеток і відстеження пакунків.

Налаштування сповіщень: Налаштуйте сповіщення електронною поштою, щоб інформувати користувачів про важливі події, такі як підтвердження замовлень, оновлення доставки та сповіщення, пов'язані з обліковим записом. Налаштуйте шаблони електронних листів, щоб вони відповідали стилю і тону інтернет-магазину. Налаштуйте сповіщення для адміністраторів, щоб вони отримували повідомлення про нові замовлення, рівень запасів та системні помилки.

Налаштування пошуку та фільтрації: Налаштуйте функціонал пошуку та фільтрації в системі, щоб покращити користувацький досвід. Налаштуйте алгоритми пошуку та рейтинг релевантності, щоб отримати точні та змістовні результати пошуку. Налаштуйте розширені фільтри на основі таких атрибутів, як автор, жанр, дата випуску або ціновий діапазон, щоб допомогти користувачам звузити пошук.

Налаштування інтеграції: Налаштуйте інтеграцію із зовнішніми системами або API. Це може включати інтеграцію зі сторонніми сервісами для обробки платежів, доставки або аналітики. Налаштуйте ключі API і механізми автентифікації, щоб встановити безпечний і надійний зв'язок з цими сервісами. Протестуйте інтеграцію, щоб забезпечити безперебійний обмін даними.

Оптимізація продуктивності: Налаштуйте продуктивність системи за допомогою механізмів кешування, індексів бази даних та оптимізації на стороні сервера. Впровадьте кешування для даних, до яких часто звертаються, щоб

зменшити навантаження на базу даних і покращити час відгуку. Налаштуйте параметри сервера, такі як таймаути, ліміти з'єднань і розподіл ресурсів, щоб максимізувати продуктивність.

Локалізація та мовні налаштування: Налаштуйте систему на підтримку декількох мов і параметрів локалізації. Налаштуйте мовні опції та забезпечте переклад тексту, призначеного для користувача, повідомлень про помилки та системних підказок. Налаштуйте формати дати та часу, символи валют та інші регіональні параметри відповідно до цільової аудиторії.

Резервне копіювання та відновлення: Налаштуйте автоматичні процедури резервного копіювання для захисту критично важливих даних. Налаштуйте частоту резервного копіювання, періоди зберігання та місця зберігання. Протестуйте процес резервного копіювання та відновлення, щоб переконатися, що дані можуть бути відновлені у разі втрати даних або збою системи. Регулярно перевіряйте журнали резервного копіювання та періодично проводьте тренування з відновлення.

Оновлення та обслуговування системи: Налагодьте процес застосування оновлень програмного забезпечення, патчів безпеки та виправлень помилок. Регулярно перевіряйте наявність оновлень для компонентів системи, включаючи серверну програму, веб-сервер, систему управління базами даних та будь-які сторонні бібліотеки. Заплануйте вікна планового технічного обслуговування для виконання оновлення та оптимізації системи.

Налаштування та уподобання користувачів: Дозвольте користувачам налаштовувати свої облікові записи та уподобання. Надайте можливості для керування профілем, сповіщень електронною поштою, мовних налаштувань та збережених способів оплати. Дозвольте користувачам персоналізувати свій досвід, обираючи улюблені жанри або авторів, створюючи списки бажань або зберігаючи історію переглядів.

4.4 План тестування

Комплексний план тестування необхідний для того, щоб система управління каталогом коміксів функціонувала належним чином і відповідала визначеним вимогам. План тестування повинен охоплювати різні аспекти системи, включаючи функціональність, продуктивність, безпеку і зручність використання.

Модульне тестування:

Виконуйте модульні тести окремих компонентів, таких як функції, класи або модулі, щоб переконатися, що вони працюють коректно в ізоляції.

Тестуйте граничні ситуації, обробляйте винятки та перевіряйте очікувану поведінку кожного компонента.

Використовуйте фреймворки для тестування, такі як Mocha, Jest або Jasmine, щоб автоматизувати і спростити процес модульного тестування.

Інтеграційне тестування:

Проводьте інтеграційні тести для перевірки взаємодії між різними компонентами системи, такими як веб-сервер, база даних і зовнішні сервіси.

Тестуйте кінцеві точки API та потоки даних, щоб забезпечити безперебійну інтеграцію між додатком на стороні сервера та інтерфейсами на стороні клієнта.

Перевірка узгодженості та цілісності даних між різними компонентами та підсистемами.

Функціональне тестування:

Проведіть функціональне тестування для перевірки основних функціональних можливостей системи.

Тестуйте такі сценарії, як додавання коміксів до каталогу, пошук коміксів, оформлення замовлень та керування акаунтами користувачів.

Переконайтеся, що система поводить себе так, як очікується, і дає бажані результати.

Протестуйте різні ролі та дозволи користувачів, щоб переконатися, що контроль доступу застосовується правильно.

Тестування продуктивності:

Проведіть тестування продуктивності, щоб оцінити швидкість реакції та стабільність системи за різних умов навантаження.

Імітуйте реальний трафік користувачів за допомогою інструментів навантажувального тестування, таких як Apache JMeter або Gatling.

Вимірюйте час відгуку, пропускну здатність і використання ресурсів, щоб виявити вузькі місця в продуктивності.

Оптимізуйте запити до бази даних, впроваджуйте механізми кешування або збільшуйте ресурси сервера для підвищення продуктивності.

Тестування безпеки:

Виконуйте тестування безпеки для виявлення вразливостей і забезпечення стійкості системи до атак.

Проведіть тестування на проникнення, щоб виявити потенційні лазівки в безпеці та перевірити ефективність заходів безпеки.

Тестування на наявність загальних ризиків безпеки, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF).

Впроваджуйте безпечні методи кодування, перевірку вхідних даних і протоколи шифрування, щоб зменшити ризики безпеки.

Юзабіліті-тестування:

Проведіть юзабіліті-тестування, щоб оцінити зручність і доступність системи.

Залучіть репрезентативних користувачів для навігації в системі та надання зворотного зв'язку щодо інтерфейсу, навігаційного потоку та загального користувацького досвіду.

Визначте сфери для вдосконалення та внесіть необхідні корективи для покращення зручності використання.

Забезпечити відповідність стандартам доступності для користувачів з обмеженими можливостями.

Тестування сумісності:

Проведіть тестування на сумісність, щоб переконатися, що система безперебійно працює в різних веб-браузерах, операційних системах і на різних пристроях.

Протестуйте систему в таких популярних браузерах, як Chrome, Firefox, Safari та Edge, враховуючи різні версії.

Перевірте адаптивність і узгодженість макетів на різних розмірах екранів і роздільних здатностях, включаючи настільні комп'ютери, планшети і мобільні пристрої.

Регресійне тестування:

Проводьте регресійне тестування щоразу після внесення змін або оновлень системи, щоб переконатися, що існуючі функції не постраждали.

Повторно запускайте раніше виконані тести, щоб підтвердити, що система все ще працює належним чином після модифікацій.

Перевіряйте виправлення помилок, щоб переконатися, що вони не призводять до нових проблем або регресу.

Тестування прийнятності для користувачів (UAT):

Залучайте кінцевих користувачів або зацікавлені сторони до проведення тестування прийнятності для користувачів.

Дозвольте користувачам протестувати систему в реальному середовищі та надати зворотній зв'язок щодо її зручності, функціональності та загального задоволення.

Розглядайте відгуки користувачів і вносьте необхідні корективи на основі їхніх пропозицій.

Перевірка документації:

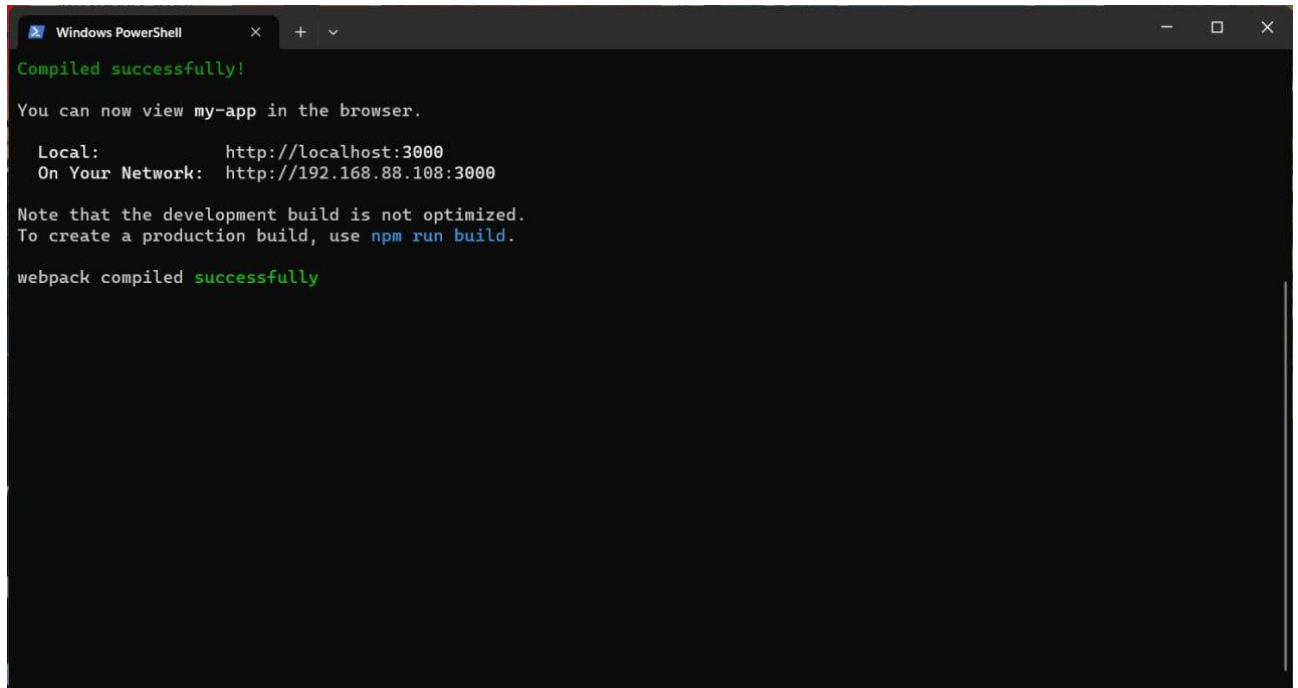
Перевірте точність і повноту системної документації, в тому числі посібників користувача, інструкцій з встановлення та інструкцій з усунення несправностей.

Переконайтеся, що документація точно відображає фактичну поведінку та конфігурацію системи.

4.5.Результати тестування розробленого додатку

Після дослідження матеріалу встановлення всіх необхідних програм та бібліотек та розробивши програмний застосунок переходимо до його тестування.

Запускаємо спочатку сервер (рис. 4.1)



```
Windows PowerShell
Compiled successfully!

You can now view my-app in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.88.108:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Рис. 4.1. Запуск серверу на системі

Далі перехід буде на головну сторінку сайту (рис. 4.2). Меню необхідне нам для перевірки чи на сайт додуються матеріали

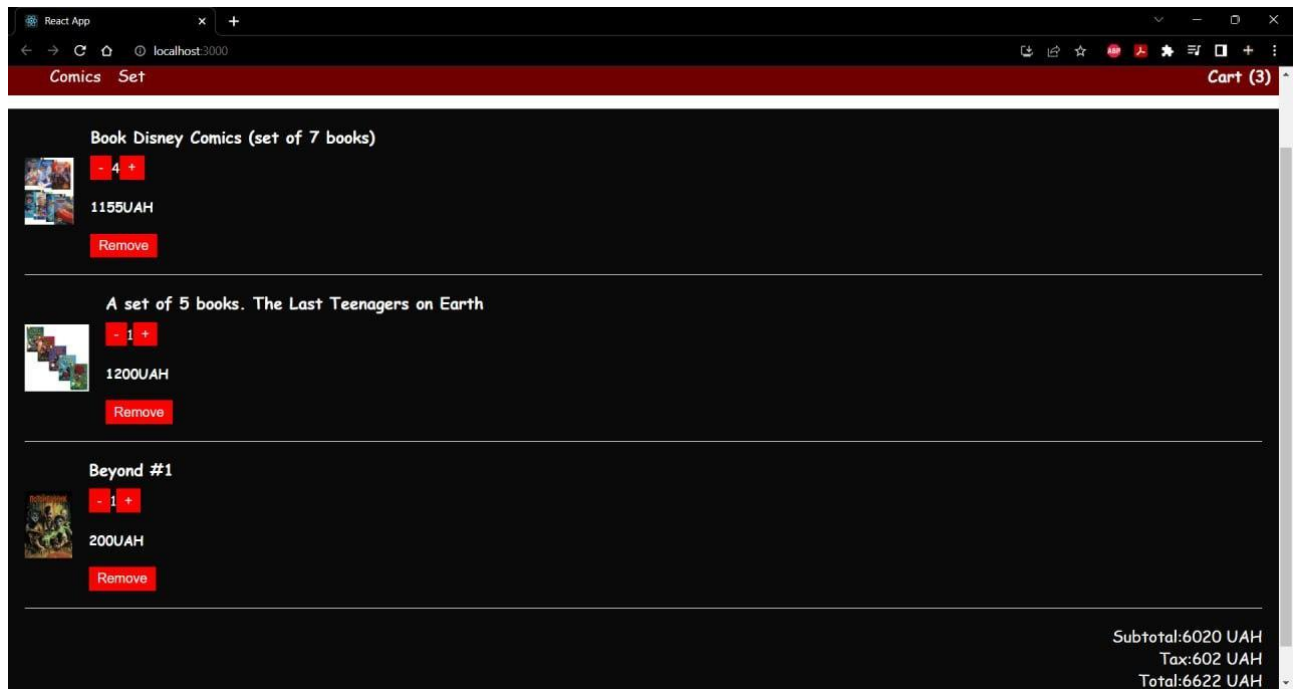


Рис. 4.5. Видалення коміксів з бази

Перевіряючи всі функції зроблено висновок що додаток готовий оскільки він виконує всі покладені на нього функції.

У майбутньому є сенс зробити з нього інтернет магазин оскільки передбачено інтеграція платіжних систем.

ВИСНОВКИ

Розробка та розгортання системи управління каталогом коміксів включала різні етапи, зокрема аналіз вимог, проектування, реалізацію, тестування та конфігурацію. На основі цих етапів можна зробити кілька ключових висновків і висновків.

Аналіз вимог: Ретельний аналіз вимог має вирішальне значення для забезпечення чіткого розуміння призначення системи, її функціональності та очікувань користувачів. Він допомагає створити міцний фундамент для процесу розробки та забезпечує узгодження з потребами зацікавлених сторін.

Розробка та реалізація: Система була розроблена та реалізована з використанням JavaScript як основної мови програмування. Інтерфейс на стороні клієнта був побудований з використанням фреймворків HTML, CSS та JavaScript, тоді як серверний додаток використовував Node.js та систему управління базами даних. Ця комбінація забезпечила масштабовану та ефективну архітектуру для виконання операцій з управління каталогом.

Тестування: Для забезпечення надійності, функціональності та продуктивності системи було проведено ретельне тестування, включаючи модульне тестування, інтеграційне тестування, функціональне тестування, тестування продуктивності, тестування безпеки, юзабіліті-тестування, тестування сумісності та регресійне тестування. Тестування допомогло виявити та вирішити проблеми і забезпечило впевненість у стабільності та коректності роботи системи.

Конфігурація: Конфігурація системи включала налаштування адміністративних параметрів, конфігурацію каталогу, налаштування цін та оплати, конфігурацію доставки, налаштування сповіщень, налаштування пошуку та фільтрації, налаштування інтеграції, оптимізацію продуктивності, локалізацію та мовні налаштування, резервне копіювання та відновлення, а також налаштування та уподобання користувачів. Належна конфігурація

забезпечила адаптацію системи до конкретних вимог та безперебійну роботу користувачів.

Розгортання: Система була успішно розгорнута на обраній серверній інфраструктурі, з встановленими та налаштованими необхідними програмними компонентами. Для захисту системи від загроз були впроваджені заходи безпеки, включаючи SSL-сертифікати, брандмауери та засоби контролю доступу. Процеси моніторингу та обслуговування були створені для забезпечення стабільності та продуктивності системи.

Задоволеність користувачів: Відгуки користувачів та їхня участь у фазах тестування та прийняття користувачами відіграли життєво важливу роль у формуванні зручності використання системи та загального рівня задоволеності. Постійні вдосконалення, засновані на відгуках користувачів, допомогли покращити функціональність системи та користувацький досвід.

Документація: Ретельна документація, включаючи системні вимоги, проектні специфікації, інструкції з розгортання та посібники користувача, була створена для полегшення розуміння системи, обслуговування та усунення несправностей. Добре задокументованими системами легше керувати і підтримувати в довгостроковій перспективі.

Отже, розробка, розгортання та налаштування системи управління каталогом коміксів вимагали ретельного планування, уваги до деталей і суворого тестування. Успішне впровадження системи та дотримання вимог користувачів і найкращих галузевих практик забезпечують надійне та ефективне рішення для управління каталогом коміксів в інтернет-магазині. Постійний моніторинг, обслуговування та зворотній зв'язок з користувачами забезпечать подальший успіх і вдосконалення системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Gamble, J., & Blackwell, R. (2001). Knowledge-Based Systems. London: Thomson Learning.
2. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. New York: McGraw-Hill Education.
3. Sommerville, I. (2016). Software Engineering. Boston: Pearson Education.
4. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Boston: Addison-Wesley.
5. Sommerville, I. (2015). Requirements Engineering: A Good Practice Guide. Chichester: John Wiley & Sons.
6. Martin, R. C. (2002). Agile Software Development: Principles, Patterns, and Practices. Boston: Prentice Hall.
7. Fowler, M., & Highsmith, J. (2001). The Agile Manifesto: Principles Behind the Agile Manifesto. Retrieved from <http://agilemanifesto.org/>
8. Kruchten, P. (2004). The Rational Unified Process: An Introduction (3rd ed.). Reading: Addison-Wesley.
9. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley.
10. Brooks, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering. Reading: Addison-Wesley.
11. McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction. Redmond: Microsoft Press.
12. Martin, R. C. (2009). Clean Code: A Handbook of Agile Software Craftsmanship. Boston: Prentice Hall.
13. Hunt, A., & Thomas, D. (2000). The Pragmatic Programmer: Your Journey to Mastery. Boston: Addison-Wesley.
14. Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Boston: Addison-Wesley.

- 15.Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Boston: Addison-Wesley.
- 16.Beck, K. (1999). Extreme Programming Explained: Embrace Change. Boston: Addison-Wesley.
- 17.Schwaber, K. (2004). Agile Project Management with Scrum. Redmond: Microsoft Press.
- 18.Cohn, M. (2004). User Stories Applied: For Agile Software Development. Boston: Addison-Wesley.
- 19.Cockburn, A. (2006). Agile Software Development: The Cooperative Game (2nd ed.). Boston: Addison-Wesley.
- 20.Williams, L., & Kessler, R. (2001). Pair Programming Illuminated. Boston: Addison-Wesley.

ДОДАТКИ

Додаток А

App.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
  'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
  sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
  monospace;
}
body:not(.navbar):not(.header) {
  font-family: 'Magra', cursive;
}
@charset "UTF-8";
.container {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #0a0a0a;
  color: white;
}
.header {
  font-family: 'Nova Round', cursive;
  background-color: #0a0a0a;
  color: white;
}
.product-row {
  display: flex;
```

```

        flex-wrap: wrap;
        margin-bottom: 1rem;
    }
    .product-card {
        flex-basis: calc(25% - 1rem);
        margin-right: 1rem;
        margin-bottom: 1rem;
    }
    .product-card:last-child {
        margin-right: 0;
    }
    .product-image {
        width: 100%;
    }
    @media (max-width: 768px) {
        .product-card {
            flex-basis: calc(50% - 1rem);
        }
    }
    .navbar {
        font-family: 'Nova Round', cursive;
        background-color: #700000;
        color: white;
    }
    .navbar li {
        display: inline-block;
        padding: 10px;
        cursor: pointer;
        font-size: 1.2rem;
    }
    .navbar li.active {
        background-color: red;
    }
    .cart {
        float: right;
        font-weight: bold;
    }
    .cart:hover {
        background-color: red;
    }

```

```
}  
.main {  
  background-color: #0a0a0a;  
  color: white;  
  padding: 20px;  
}  
.cart-item {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  margin-bottom: 20px;  
  padding-bottom: 20px;  
  border-bottom: 1px solid white;  
}  
.cart-item img {  
  max-height: 80px;  
  object-fit: contain;  
}  
.cart-item-details {  
  flex-grow: 1;  
  margin-left: 20px;  
}  
.cart-item-details h3 {  
  margin-top: 0;  
  margin-bottom: 10px;  
  font-size: 1.2rem;  
}  
.cart-item-details button {  
  border: none;  
  padding: 5px 10px;  
  background-color: red;  
  color: white;  
  cursor: pointer;  
  font-size: 1rem;  
}  
.cart-item-details button:hover {  
  background-color: #700000;  
}  
.cart-item-quantity {
```

```

    display: flex;
    align-items: center;
}

.cart-item-quantity button {
    border: none;
    padding: 5px 10px;
    background-color: red;
    color: white;
    cursor: pointer;
    font-size: 1rem;
}

.cart-item-quantity button:hover {
    background-color: #700000;
}

.cart-total {
    text-align: right;
    font-size: 1.2rem;
    margin-top: 20px;
    color: white; /* added to change the text color */
}

.tax {
    font-size: 1.2rem;
    margin-top: 20px;
    text-align: right;
    color: white; /* added to change the text color */
}

.form {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 400px;
    padding: 20px;
    background-color: #0a0a0a;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
}

.input{
    color: #333333;

```

```

}
.input:focus {
    outline: none;
    border-color: #4CAF50;
}
.success-message {
    margin-top: 10px;
    color: #4CAF50;
}
.error-message {
    margin-top: 10px;
    color: red;
}
@media screen and (max-width: 768px) {
    .navbar li {
        display: block;
        padding: 10px;
        text-align: center;
    }
    .cart {
        float: none;
        margin-top: 10px;
    }
    .cart-item {
        flex-direction: column;
        text-align: center;
    }

    .cart-item img {
        margin-bottom: 10px;
    }
    .cart-item-details {
        margin: 0;
    }
    .cart-item-quantity {
        margin-top: 10px;
        justify-content: center;
    }
    .form {

```

```
        width: 100%;  
        padding: 10px;  
    }  
}
```

App.js

```

import React, { useState } from 'react';
import './App.css';
import { Comics } from './data';
import { Set } from './data2';
function App() {
  const [cartItems, setCartItems] = useState([]);
  const [activeTab, setActiveTab] = useState('comics');
  const [newProduct, setNewProduct] = useState({
    title: "",
    price: "",
    img: "",
    info: ""
  });
  const [products, setProducts] = useState(Comics);
  const [packages, setPackages] = useState(Set);
  const handleInputChange = (event) => {
    const { name, value } = event.target;
    setNewProduct({ ...newProduct, [name]: value });
  };
  const handleFormSubmit = (event) => {
    event.preventDefault();
    const newId = products.length + 1;
    const newProductObj = {
      id: newId,
      title: newProduct.title,
      price: parseInt(newProduct.price),
      img: newProduct.img,
      info: newProduct.info
    };
    if (activeTab === "comics") {
      setProducts([...products, newProductObj]);
    } else if (activeTab === "set") {
      setPackages([...packages, newProductObj]);
    }
    setNewProduct({
      title: "",
      price: "",

```

```

        img: "",
        info: ""
    });
};

const addProductToCart = (product) => {
    const existingProductIndex = cartItems.findIndex((item) =>
item.id === product.id);
    if (existingProductIndex !== -1) {
        const updatedCartItems = [...cartItems];
        updatedCartItems[existingProductIndex].quantity += 1;
        setCartItems(updatedCartItems);
    } else {
        const updatedCartItems = [...cartItems, { ...product,
quantity: 1 }];
        setCartItems(updatedCartItems);
    }
};

const removeProductFromCart = (productId) => {
    const existingProductIndex = cartItems.findIndex((item) =>
item.id === productId);
    if (existingProductIndex !== -1) {
        const updatedCartItems = [...cartItems];
        updatedCartItems.splice(existingProductIndex, 1);
        setCartItems(updatedCartItems);
    }
};

const increaseProductQuantity = (productId) => {
    const updatedCartItems = cartItems.map((product) => {
        if (product.id === productId) {
            return { ...product, quantity: product.quantity + 1 };
        }
        return product;
    });
    setCartItems(updatedCartItems);
};

const decreaseProductQuantity = (productId) => {
    const updatedCartItems = cartItems.map((product) => {
        if (product.id === productId) {
            const updatedQuantity = product.quantity - 1;

```

```

        if (updatedQuantity <= 0) {
            return null;
        }
        return { ...product, quantity: updatedQuantity };
    }
    return product;
  }).filter((product) => product !== null);
  setCartItems(updatedCartItems);
};

const getCartSubtotal = () => {
  return cartItems.reduce(
    (total, product) => total + product.price * product.quantity,
    0);
};

const getCartTax = () => {
  return getCartSubtotal() * 0.1;
};

const getCartTotal = () => {
  return getCartSubtotal() + getCartTax();
};

const clearCart = () => {
  setCartItems([]);
};

const [showProduct, setShowProduct] = useState(null);
const handleProductHover = (product) => {
  setShowProduct(product);
};

const handleProductLeave = () => {
  setShowProduct(null);
};

const renderProductInfo = (product) => {
  if (showProduct && showProduct.id === product.id) {
    return (
      <div className="product-info">
        <p>{product.info}</p>
      </div>
    );
  }
};

```

```

const renderProducts = (products) => {
  const rows = [];
  for (let i = 0; i < products.length; i += 4) {
    rows.push(
      <div className="product-row" key={i}>
        {products.slice(i, i + 4).map((product) => (
          <div
            key={product.id}
            className="product-card"
            onMouseEnter={() =>
handleProductHover(product)}
            onMouseLeave={() => handleProductLeave()}
          >
            <img
              src={product.img}
              alt={product.title}
              className="product-image"
            />
            <h3>{product.title}</h3>
            <h4>{product.price} UAH</h4>
            <button onClick={() =>
addProductToCart(product)}>Add to Cart</button>
            {renderProductInfo(product)}
          </div>
        ))}
      </div>
    );
  }
  return rows;
};

const renderCartItems = () => {
  return cartItems.map((product) => (
    <div key={product.id} className="cart-item">
      <img
        src={product.img}
        alt={product.title}
        className="cart-item-image"
      />
      <div className="cart-item-details">

```

```

        <h3>{product.title}</h3>
        <div className="cart-item-quantity">
            <button onClick={() =>
decreaseProductQuantity(product.id)}>-</button>
            <span>{product.quantity}</span>
            <button onClick={() =>
increaseProductQuantity(product.id)}>+</button>
        </div>
        <h4>{product.price}UAH</h4>
        <button onClick={() =>
removeProductFromCart(product.id)}>Remove</button>
    </div>
</div>
));
};
const renderCart = () => {
    return (
        <>
            <div className="cart-items">
                {renderCartItems()}
                <div className="cart-total">
                    <div className="cart-total-line">
                        <span>Subtotal:</span>
                        <span>{getCartSubtotal()} UAH</span>
                    </div>
                    <div className="cart-total-line">
                        <span>Tax:</span>
                        <span>{getCartTax()} UAH</span>
                    </div>
                    <div className="cart-total-line">
                        <span>Total:</span>
                        <span>{getCartTotal()} UAH</span>
                    </div>
                </div>
            </div>
            <div className="cart-buttons">
                <button onClick={() => clearCart()}>Clear
Cart</button>

```

```

        <button onClick={() => alert('Checkout not
implemented yet!')}>
            Checkout
        </button>
    </div>
</>
);
};
return (
    <div className="app">
        <header className="header">
            <h1 className="store-name">ComicZone</h1>
        </header>

        <nav className="navbar">
            <ul>
                <li
                    className={activeTab === 'comics' ? 'active' :
''}

                    onClick={() => setActiveTab('comics')}
                >
                    Comics
                </li>
                <li
                    className={activeTab === 'set' ? 'active' : ''}
                    onClick={() => setActiveTab('set')}
                >
                    Set
                </li>
                <li className="cart" onClick={() =>
setActiveTab('cart')}>
                    Cart ({cartItems.length})
                </li>
            </ul>
        </nav>
        <main className="main">
            {activeTab === 'comics' && (
                <div className="products">{renderProducts(products)}
                <form onSubmit={handleFormSubmit}>

```

```

        <h2>Add a New Product</h2>
        <label htmlFor="title">Title:</label>
        <input type="text" id="title" name="title"
value={newProduct.title} onChange={handleInputChange} />
        <label htmlFor="price">Price:</label>
        <input type="text" id="price" name="price"
value={newProduct.price} onChange={handleInputChange} />
        <label htmlFor="info">Info:</label>
        <textarea id="info" name="info"
value={newProduct.info} onChange={handleInputChange}></textarea>
        <label htmlFor="img">URL img:</label>
        <input type="text" id="img" name="img"
value={newProduct.img} onChange={handleInputChange} />
        <button type="submit">Add Product</button>
    </form>
</div>
)}
{activeTab === 'set' && (
    <div className="products">{renderProducts(packages)}
        <form onSubmit={handleFormSubmit}>
            <h2>Add a New Product</h2>
            <label htmlFor="title">Title:</label>
            <input type="text" id="title" name="title"
value={newProduct.title} onChange={handleInputChange} />
            <label htmlFor="price">Price:</label>
            <input type="text" id="price" name="price"
value={newProduct.price} onChange={handleInputChange} />
            <label htmlFor="info">Info:</label>
            <textarea id="info" name="info"
value={newProduct.info} onChange={handleInputChange}></textarea>
            <label htmlFor="img">URL img:</label>
            <input type="text" id="img" name="img"
value={newProduct.img} onChange={handleInputChange} />
            <button type="submit">Add Product</button>
        </form>
    </div>
)}
{activeTab === 'cart' && renderCart()}
</main>

```

```
        </div>
    );
}
export default App;
```