НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

інформаційних систем і технологій

_____ Швиденко М.З., доц., к.е.н.

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Проектування та розробка веб-орієнтованої системи для електронної комерції»

Спеціальність 122 «Комп'ютерні науки»

Гарант освітньої програми

к.т.н. доцент _____ Руденський Р.А.

(Науковий ступень та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

д-р філос., доц. _____ Корольчук В.І.

(Науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ Мирошниченко Д.І.

(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформаційних систем і технологій
_____ Швиденко М.З., доц., к.е.н.
(підпис) (ПІБ, вчене звання і ступінь)
«___»_____2025 р

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

Мирошниченко Данило Ігорович

Спеціальність 122 – «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Проектування та розробка веб-орієнтованої системи для електронної комерції» затверджена наказом ректора НУБІП України від 16.12.2024 №2246 «С»
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані до роботи: опис програмного забезпечення
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "___"_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи

_____д-р філос., доц.____ _____ ___Корольчук В.І._
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Мирошниченко Д.І.
(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SPA - Single Page Application

SWOT - Strengths (сильні сторони), Weaknesses (слабкі сторони), Opportunities (можливості) та Threats (загрози)

UML - Unified Modeling Language

СУБД - системи управління базами даних

ВСТУП

У сучасному світі інформаційних технологій електронна комерція стала одним із ключових драйверів розвитку цифрової економіки. Зростання кількості онлайн-користувачів, поширення мобільних пристроїв, підвищення швидкості доступу до Інтернету та зміна споживчих звичок стимулюють розвиток веб-орієнтованих платформ для продажу товарів і послуг. Електронна комерція не лише відкриває нові можливості для підприємництва, але й підвищує конкуренцію, вимагаючи від бізнесу впровадження сучасних, інноваційних рішень у сфері інформаційних технологій.

Актуальність теми обумовлена необхідністю створення гнучких, масштабованих та безпечних веб-систем, здатних забезпечити повноцінне функціонування електронного магазину, з урахуванням потреб як користувачів, так і адміністраторів. Незважаючи на наявність багатьох готових платформ, попит на індивідуальні рішення, адаптовані під конкретні бізнес-процеси, залишається високим.

Метою даної дипломної роботи є проєктування та розробка веб-орієнтованої системи електронної комерції, яка забезпечує управління каталогом товарів, обробку замовлень, облік користувачів та реалізацію платіжних модулів. Основну увагу приділено архітектурі системи, зручності користувацького інтерфейсу.

Об'єктом дослідження є процеси розробки веб-додатків для електронної комерції.

Предметом – методи проєктування архітектури, технології розробки інтерфейсів та організації взаємодії з базами даних.

Для реалізації поставленої мети в роботі вирішуються такі завдання:

- аналіз сучасних рішень у сфері e-commerce;

- проєктування структури веб-системи з урахуванням вимог безпеки та масштабованості;
- реалізація функціонального прототипу з використанням актуальних веб-технологій;
- тестування системи та формулювання рекомендацій щодо її подальшого вдосконалення.

Результати дослідження можуть бути використані як основа для впровадження власного онлайн-магазину або адаптовані для потреб малого та середнього бізнесу.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Електронна комерція, або e-commerce, – це форма торговельної діяльності, що здійснюється з використанням електронних каналів зв’язку, насамперед Інтернету. Сьогодні вона охоплює не лише онлайн-магазини, а й маркетплейси, цифрові сервіси, мобільні застосунки та навіть цілі екосистеми бізнесу в мережі. В умовах цифрової трансформації бізнесу веборієнтовані системи стали ключовим інструментом для автоматизації операцій, покращення обслуговування клієнтів та аналітики бізнес-процесів. Вони дозволяють компаніям ефективно використовувати великі дані, соціальні медіа та аналітичні інструменти для підвищення продуктивності, стійкості та конкурентоспроможності, особливо в B2B-сегменті [1], [2]. Веборієнтовані платформи сприяють інноваціям у продуктах, процесах і бізнес-моделях, дозволяючи компаніям швидко адаптуватися до змін ринку та впроваджувати нові цифрові стратегії [3]. Важливу роль відіграють такі технології, як Інтернет речей, хмарні обчислення та прогнозна аналітика, які трансформують дані у цінну інформацію для прийняття рішень [4].

Для успішної цифрової трансформації компаніям рекомендується поступово впроваджувати нові технології, розвивати партнерства, стандартизувати процеси та приділяти увагу етиці даних [3]. Водночас, цифрова трансформація вимагає перегляду організаційних структур, культури та управлінських підходів, а також активної участі керівництва [3]. Дослідження також підкреслюють важливість інтеграції компаній у цифрові екосистеми та платформенні рішення, що розширює можливості для співпраці та інновацій [5]. Для малих бізнесів особливо важливою є підтримка з боку держави у вигляді цифрових платформ, навчання та стимулювання співпраці7. Загалом,

веборієнтовані системи є фундаментом сучасної цифрової трансформації, забезпечуючи гнучкість, масштабованість і стійкість бізнесу [5].

Завдяки цьому компанії отримують змогу підвищувати свою конкурентоспроможність, ефективніше працювати з цільовою аудиторією та знижувати операційні витрати.

Предметна область електронної комерції охоплює складну, але добре структуровану взаємодію між кількома ключовими категоріями користувачів. Адміністратори системи є "диригентами" онлайн-магазину, відповідальними за його безперебійне функціонування та актуальність контенту. Їхні обов'язки включають не лише редагування товарів і категорій (додавання нових позицій, оновлення описів, цін та зображень), але й управління замовленнями на всіх етапах – від їхнього надходження до відправки та скасування. Крім того, адміністратори здійснюють моніторинг дій користувачів для виявлення потенційних проблем або зловживань, а також загальну підтримку працездатності системи, що включає технічне обслуговування, оновлення та виправлення помилок.

Зареєстровані клієнти є ядром цільової аудиторії, яким доступний повний цикл взаємодії з системою. Це охоплює не тільки перегляд товарів та їх детальне вивчення, а й можливість формування замовлень, зручне здійснення оплати через інтегровані платіжні системи та перегляд історії покупок, що покращує користувацький досвід та спрощує повторні замовлення. Наявність особистого кабінету дозволяє їм зберігати персональні дані, адреси доставки та налаштування, що значно прискорює процес оформлення замовлень.

Гостьові (незареєстровані) відвідувачі, хоч і обмежені у функціоналі, відіграють важливу роль як потенційні клієнти. Їм доступний функціонал перегляду товарів, пошук та ознайомлення з асортиментом, що дозволяє їм вивчити пропозицію без необхідності негайної реєстрації. Однак, для того, щоб оформити замовлення, вони обов'язково повинні пройти реєстрацію або авторизацію. Це забезпечує системі необхідні дані для обробки замовлення та

подальшої комунікації з клієнтом, а також дозволяє відстежувати їхні покупки та будувати персоналізовані пропозиції в майбутньому.

В межах предметної області електронної комерції реалізується низка важливих функціональних складових, кожна з яких є невід'ємною частиною успішного онлайн-бізнесу. Однією з найфундаментальніших є каталог товарів. Він не просто відображає асортимент, а й повинен підтримувати ієрархічну структуру (наприклад, категорії та підкатегорії), що дозволяє користувачам легко орієнтуватися серед тисяч позицій. Можливості фільтрації (за ціною, брендом, характеристиками) та пошуку (за ключовими словами) є критично важливими для швидкого знаходження потрібних товарів, а відображення розширеної інформації про кожен товар (детальні описи, технічні характеристики, відгуки, фотографії та відео) допомагає покупцям прийняти обґрунтоване рішення.

Система кошика є центральним елементом процесу покупки, дозволяючи користувачам зберігати обрані позиції перед оформленням замовлення, коригувати кількість товарів та переглядати загальну вартість. Цей модуль повинен бути інтуїтивно зрозумілим та надійним. Модуль оплати та доставки забезпечує зручний і захищений механізм фінансових транзакцій (інтеграція з різними платіжними системами, можливість оплати карткою, через електронні гаманці тощо) та логістичних операцій (вибір методів доставки, розрахунок вартості, відстеження статусу замовлення). Надійність цих модулів є запорукою довіри клієнтів.

Не менш важливою складовою є адміністративна панель, яка виступає ядром керування веборієнтованою системою електронної комерції. Це потужний інструмент, що дозволяє ефективно координувати всі елементи системи: від додавання нових товарів, редагування їхніх характеристик і завантаження зображень – до повного контролю за наявністю, цінами, категоріями та акційними пропозиціями. Окрім управління товарним асортиментом, адміністратор має змогу керувати обліковими записами користувачів, змінювати

їхні ролі (наприклад, активувати або деактивувати обліковий запис), а також переглядати історію активності кожного з них.

Адмінпанель також забезпечує детальний моніторинг замовлень: їхній статус, суми, методи оплати, історію змін та наявність супровідної інформації. Система аналітики, інтегрована у панель керування, надає можливість переглядати статистику продажів за різні періоди, найпопулярніші товари, середній чек, динаміку нових користувачів тощо. Це дає змогу приймати обґрунтовані рішення, базуючись на актуальних даних.

Додатково передбачена функція обробки зворотного зв’язку: адміністратор може переглядати повідомлення від клієнтів, реагувати на скарги чи запити, модерацію відгуків або запитань до товарів. Добре продуманий інтерфейс адміністративної частини значно спрощує щоденну роботу персоналу, знижує ймовірність помилок і дозволяє оперативно реагувати на будь-які зміни у стані системи.

Не менш важливою є система реєстрації та авторизації користувачів, яка забезпечує безпечний доступ до персоналізованого функціоналу. Завдяки реалізації захищеної автентифікації з використанням хешування паролів і токенів, персональні дані користувачів надійно зберігаються, а доступ до конфіденційної інформації суворо обмежено. Це дозволяє користувачам безпечно входити в особистий кабінет, де вони можуть переглядати історію замовлень, редагувати профіль, керувати кошиком та відстежувати статус доставки.

Нарешті, інтеграція з платіжними шлюзами є ключовою ланкою в процесі завершення покупки. Система підтримує з'єднання з популярними сервісами онлайн-платежів, що дозволяє здійснювати транзакції в реальному часі, з миттєвим підтвердженням операцій. Така інтеграція не лише підвищує зручність для клієнта, а й забезпечує автоматичне оновлення статусу замовлень у базі даних, мінімізуючи потребу в ручному втручанні та зменшуючи ймовірність помилок при обробці оплати.

Враховуючи технічні та бізнесові вимоги до таких систем, до них висуваються жорсткі критерії щодо масштабованості, безпеки, зручності використання, мобільної адаптивності та швидкодії. Система має забезпечувати високу продуктивність навіть при значному навантаженні, гарантувати збереження конфіденційних даних користувачів за допомогою сучасних методів шифрування, а також мати зрозумілий і дружній до користувача інтерфейс. Особливої уваги потребує забезпечення доступу з різних типів пристроїв – від комп'ютерів до мобільних телефонів та планшетів, що є стандартом для сучасного онлайн-ринку.

1.2 Огляд існуючих рішень

Аналіз уже існуючих рішень є критично важливим етапом у розробці будь-якого програмного продукту, зокрема веборієнтованої системи електронної комерції. Він дозволяє виявити сильні й слабкі сторони конкурентів, уникнути типових помилок та запозичити найкращі практики у сфері дизайну, архітектури та функціоналу. Ознайомлення з наявними платформами дає змогу краще зрозуміти очікування кінцевих користувачів, а також виявити потреби, які залишаються не задоволеними поточними рішеннями. Такий аналіз значно підвищує ймовірність створення релевантного, зручного та конкурентоспроможного продукту, що має потенціал успішно функціонувати на ринку.

Існує велика кількість комерційних рішень, які вже успішно функціонують у цій сфері. Серед найвідоміших прикладів – українська платформа Rozetka, торговельний майданчик Prom.ua, міжнародні гіганти Amazon, Shopify, а також популярні CMS-платформи, як-от WooCommerce. Дослідження цих систем дозволяє виявити ключові функціональні патерни, переваги різних архітектурних підходів і визначити, які рішення можуть бути адаптовані або вдосконалені при розробці власної системи.

Однією з найпопулярніших міжнародних платформ є Shopify – комерційне SaaS-рішення, яке дозволяє швидко створити онлайн-магазин без глибоких знань програмування. Shopify (рис. 1.1) пропонує широкий спектр функцій: керування товарами, оплата, доставка, звіти, мобільна оптимізація. Проте обмеження в налаштуванні та щомісячна абонплата можуть бути недоліком для користувачів, які прагнуть повного контролю над системою.

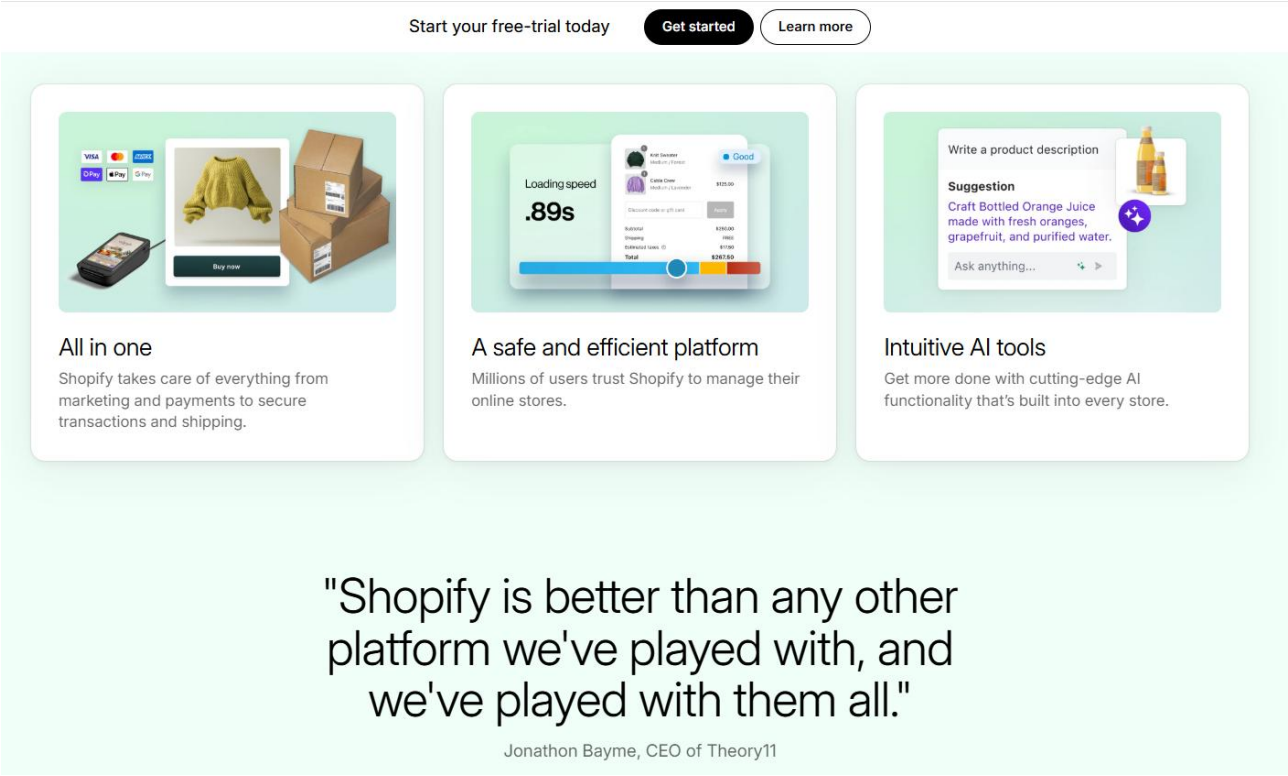

Рис. 1.1. Можливості платформи є Shopify

Інша відома платформа – WooCommerce (рис.1.2.), що є плагіном для WordPress. Це безкоштовне рішення з відкритим кодом, яке надає користувачам широку гнучкість у налаштуванні магазину, можливість використання тем і розширень, а також активну спільноту підтримки. WooCommerce добре підходить для малих і середніх бізнесів, однак при зростанні навантаження виникає потреба в оптимізації продуктивності.

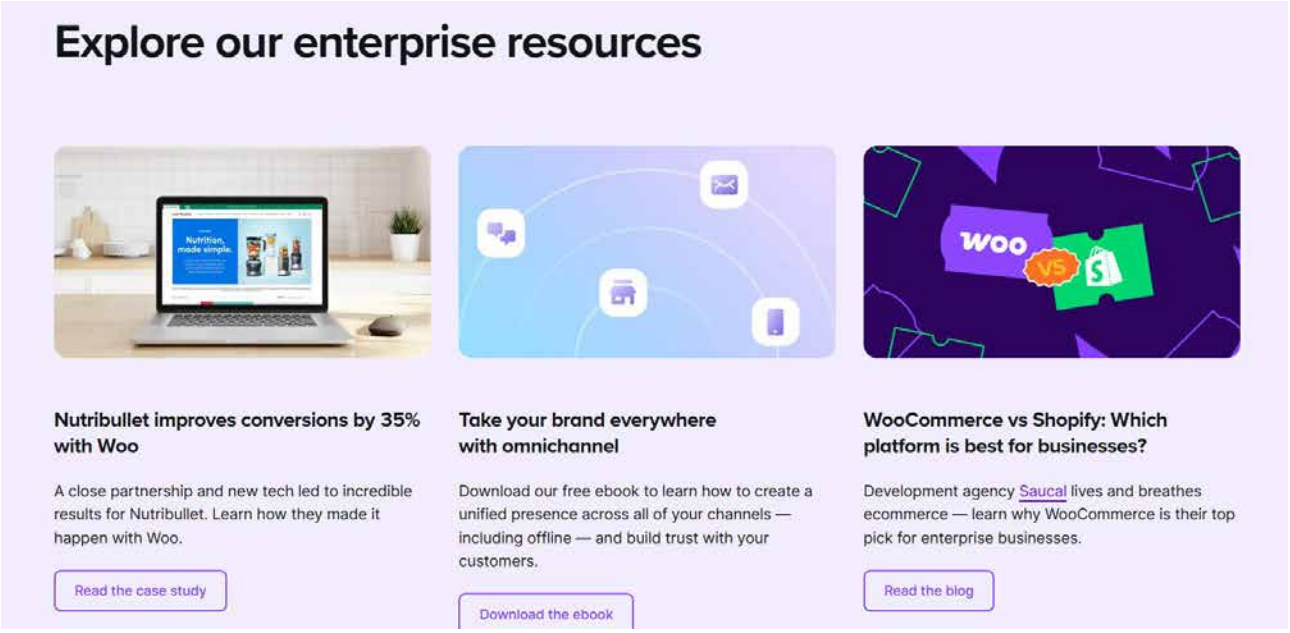

Рис. 1.2. Ресурси платформи WooCommerce

Серед українських рішень особливо варто відзначити Rozetka та Prom.ua, які виконують роль великих багатофункціональних маркетплейсів. Rozetka (рис. 1.3) є однією з найвідоміших торговельних платформ в Україні, що поєднує функціональність класичного інтернет-магазину з можливістю розміщення товарів від сторонніх продавців. Завдяки потужній технічній базі, Rozetka забезпечує стабільну роботу навіть під час пікових навантажень, а її інтерфейс оптимізований для зручного користування як на десктопних, так і на мобільних пристроях.

Платформа має розгалужену логістичну інфраструктуру, яка включає власні склади, пункти видачі та кур'єрську доставку. Це дозволяє реалізувати повний цикл обслуговування клієнта – від прийняття замовлення до його фізичного вручення покупцеві. Водночас Rozetka пропонує розширену систему фільтрації товарів, яка дозволяє покупцеві швидко знаходити потрібні позиції за безліччю параметрів – від технічних характеристик до рейтингу та ціни.

Для продавців платформа створює зручне середовище з інструментами для керування асортиментом, аналітикою, рекламними кампаніями та технічною підтримкою. Однак, незважаючи на очевидні переваги, Rozetka, як і будь-який

великий централізований сервіс, має певні обмеження: контроль за дизайном вітрини, суворі умови розміщення товарів і обмежена свобода у налаштуванні функціоналу. У цьому контексті розробка власної системи електронної комерції є доцільною для бізнесів, що прагнуть повної незалежності, кастомізації та унікального користувацького досвіду.

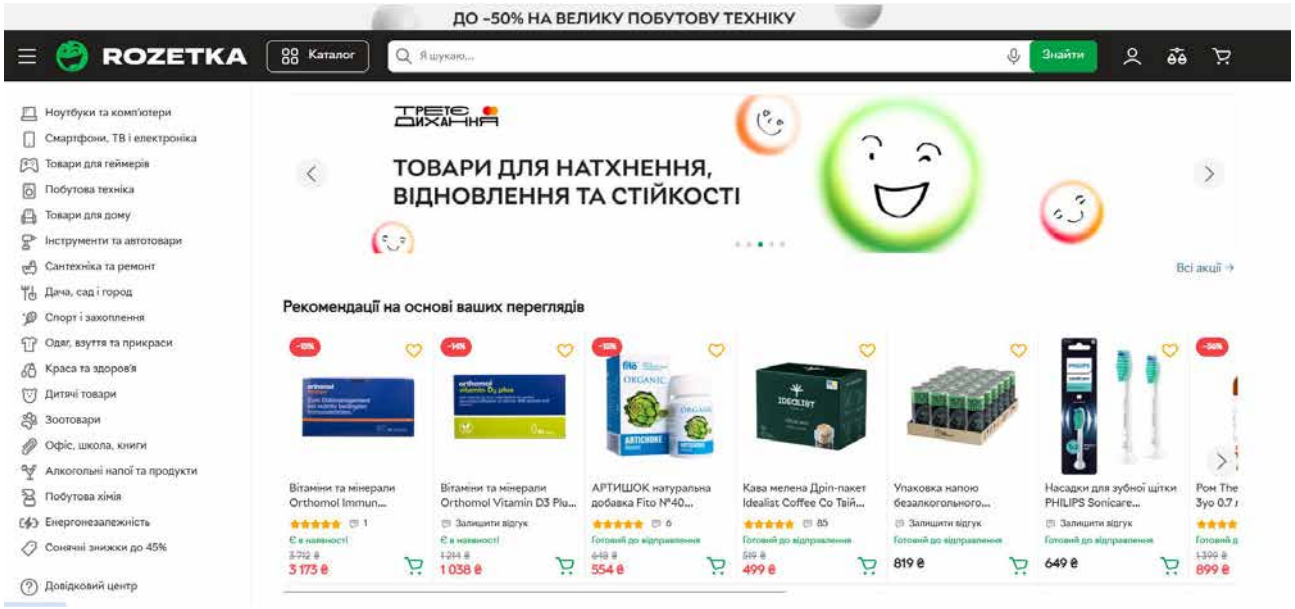

Рис.1.3. Функціонал маркетплейсу. Rozetka

Prom.ua (рис. 1.4) – одна з найпопулярніших платформ електронної комерції в Україні, яка надає користувачам інструменти для створення власних онлайн-вітрин всередині великої централізованої екосистеми. Завдяки зручному інтерфейсу та широким можливостям керування товарами, продавці можуть швидко запустити свій бізнес без необхідності глибоких технічних знань. Платформа забезпечує готові шаблони, інтеграцію з популярними службами доставки, інструменти для реклами й аналітики, а також централізовану підтримку платежів.

Однак, попри низку переваг, Prom.ua має суттєві обмеження, особливо з погляду дизайну та архітектурної гнучкості. Користувачі змушені адаптовуватись до заздалегідь визначених шаблонів, що ускладнює створення унікального брендингу чи реалізацію нестандартного функціоналу. Можливості

розширення сайту через власні модулі чи кастомні рішення обмежені, що не дозволяє масштабувати платформу під специфічні бізнес-процеси або інтегрувати сторонні системи безпосередньо. Також важливо враховувати, що продавець не має повного контролю над хостингом, швидкодією чи безпековими налаштуваннями, оскільки всі сервіси централізовано обслуговуються платформою. Це може стати критичним для середнього та великого бізнесу, який прагне мати гнучкий, незалежний та індивідуально оптимізований ресурс.

Таким чином, хоча Prom.ua добре підходить для малого бізнесу та стартових проєктів, його обмеження є однією з причин, чому власна система електронної комерції, розроблена "з нуля", може стати стратегічно вигіднішим рішенням у довгостроковій перспективі.

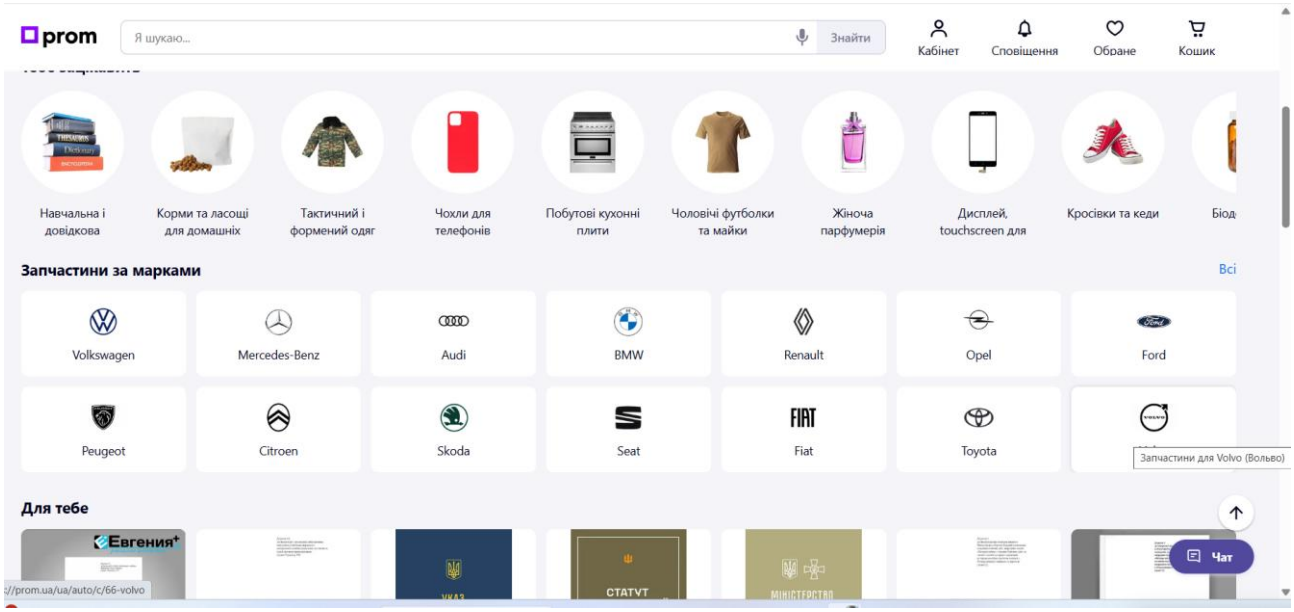

Рис. 1.4. Можливості маркетплейсу Prom.ua

Альтернативним варіантом є використання JavaScript-фреймворків, зокрема React.js або Vue.js у поєднанні з Node.js і MongoDB чи PostgreSQL. Це дозволяє створити високопродуктивний SPA (Single Page Application) із кастомною логікою й адаптованим інтерфейсом. Такий підхід передбачає повну відповідальність за архітектуру та захист даних, але забезпечує максимальну гнучкість і контроль над проєктом.

Порівнялий аналіз розглянутих інструментів представлено у таблиці 1.1.

Таблиця 1.1. Порівняльна таблиця існуючих рішень

Платформа	Тип рішення	Переваги	Недоліки
Shopify	SaaS	Швидкий запуск, підтримка, готові шаблони, платіжна інтеграція	Обмежена кастомізація, платна модель, залежність від сервісу
WooCommerce	Open Source	Безкоштовна основа, велика кількість плагінів, інтеграція з WordPress	Потребує технічної підтримки, складна оптимізація при великому навантаженні
Rozetka	Маркетплейс	Популярність, широка аудиторія, власна логістика	Обмежена індивідуалізація, конкуренція між продавцями
Prom.ua	Маркетплейс	Просте створення вітрини, український інтерфейс	Менше можливостей кастомізації, залежність від платформи
Magento	Open Source	Висока продуктивність, корпоративні функції, гнучкість	Високий поріг входу, складне адміністрування
Власна система (React/Node.js)	Індивідуальна розробка	Повний контроль, кастомна логіка, адаптивність, SPA-архітектура	Велика трудомісткість, потреба в досвідченій команді

Таким чином, аналіз існуючих рішень показує, що кожна платформа має свої переваги й обмеження. Комерційні сервіси зручні для швидкого старту, але менш гнучкі, тоді як розробка власної веборієнтованої системи дозволяє врахувати специфіку бізнесу та потреби цільової аудиторії, а також забезпечити високу адаптивність і незалежність від сторонніх провайдерів.

Для обґрунтування доцільності розробки власної веборієнтованої системи електронної комерції було проведено SWOT-аналіз (рис. 1.5), що дозволяє оцінити внутрішні сильні й слабкі сторони проєкту, а також зовнішні можливості та загрози.

Рис. 1.5. Проведений SWOT-аналіз предметної області

Такий підхід дав змогу комплексно проаналізувати перспективи реалізації системи, виявити потенційні ризики та сформувати стратегічні орієнтири для подальшої розробки.

Проведений огляд існуючих платформ електронної комерції та аналіз їхніх переваг і недоліків засвідчив, що хоча на ринку доступно багато готових рішень, жодне з них не забезпечує повного спектра можливостей для гнучкої адаптації під специфічні потреби бізнесу. Розробка власної веборієнтованої системи відкриває значні перспективи у плані кастомізації, локалізації та інтеграції з внутрішніми процесами компанії. SWOT-аналіз підтвердив наявність як сильних сторін, так і викликів, що потребують врахування під час проєктування, однак

загалом свідчить про доцільність і перспективність обраного напряму дослідження.

1.3 Аналіз вимог до програмної системи

Аналіз вимог є критично важливим етапом у розробці програмного забезпечення, оскільки саме на цьому етапі визначаються функціональні та нефункціональні характеристики майбутньої системи, а також очікування замовників і користувачів [8]. Якісний аналіз вимог дозволяє уникнути помилок у подальших фазах розробки, зменшує ризики невідповідності продукту очікуванням і сприяє ефективній комунікації між усіма зацікавленими сторонами [9]. У сучасних підходах, зокрема в гнучких (agile) методологіях, аналіз вимог інтегрується з іншими процесами, що дозволяє швидко реагувати на зміни та уточнювати вимоги протягом усього життєвого циклу проекту [10, 11]. Важливими аспектами є також пріоритизація вимог, їхня простежуваність і формалізація, що забезпечує контроль за виконанням і відповідність бізнес-цілям [12, 13].

Від правильності та повноти формулювання вимог залежить успішність реалізації функціоналу системи, її зручність для користувача та відповідність поставленим бізнес-цілям. У рамках даної дипломної роботи було визначено функціональні й нефункціональні вимоги до веборієнтованої системи електронної комерції, яка призначена для створення повноцінного онлайн-магазину з можливістю керування товарами, обробки замовлень і здійснення покупок користувачами.

Система повинна забезпечувати базову і розширену функціональність для трьох основних категорій користувачів: адміністратора, зареєстрованого користувача (покупця) та гостьового відвідувача. До ключових функціональних можливостей належать: реєстрація та авторизація користувачів; перегляд і фільтрація каталогу товарів за категоріями та параметрами; додавання товарів до кошика та оформлення замовлень; можливість онлайн-оплати; перегляд історії

покупок; для адміністратора – додавання, редагування та видалення товарів, обробка замовлень, перегляд статистики, керування користувачами.

Крім того, система повинна реалізовувати сповіщення про зміну статусу замовлення (через email або інтерфейс кабінету), а також підтримку знижок, акцій та промокодів. Особливу увагу варто приділити безпечній аутентифікації, перевірці введених даних і захисту від несанкціонованого доступу.

У контексті нефункціональних вимог система має бути масштабованою та здатною працювати в умовах значного навантаження. Однією з ключових вимог є адаптивність інтерфейсу – тобто коректне відображення на пристроях з різною роздільною здатністю: ПК, планшетах, смартфонах. Система повинна відповідати вимогам сучасної веббезпеки, зокрема підтримувати HTTPS-протокол, зберігати паролі у хешованому вигляді, мати захист від CSRF- і XSS-атак.

Також важливою є вимога щодо швидкості завантаження сторінок – навіть при великій кількості товарів чи запитів система повинна забезпечувати мінімальний час відповіді. Надійність і відмовостійкість системи забезпечуються резервним копіюванням даних та обробкою нештатних ситуацій (наприклад, відсутності з'єднання з платіжним шлюзом).

Для реалізації проєкту передбачається використання сучасного технологічного стеку. На боці клієнта – JavaScript-фреймворк (наприклад, React або Vue.js) для створення динамічного і зручного інтерфейсу. На боці сервера – Node.js з Express або інший серверний фреймворк, а для роботи з базою даних – MongoDB або PostgreSQL. Для забезпечення безпеки й автентифікації користувачів буде використано JWT або OAuth2. Важливо також передбачити можливість інтеграції з популярними платіжними сервісами, такими як LiqPay або Stripe.

Проаналізовані вимоги створюють цілісне уявлення про майбутню систему, її функціональні особливості та технічні обмеження. Такий аналіз дозволяє мінімізувати ризики під час розробки, уникнути суперечностей у

проєктуванні та забезпечити відповідність результату очікуванням як користувачів, так і бізнесу.

1.4 Моделювання предметної області

Моделювання предметної області є ключовим етапом у процесі розробки інформаційної системи, оскільки воно дозволяє формалізувати знання про бізнес-процеси, визначити основні сутності системи, їхні атрибути та взаємозв’язки. У контексті розробки веборієнтованої системи для електронної комерції необхідно ретельно описати взаємодію між користувачами, товарами, замовленнями, оплатами та іншими елементами, що беруть участь у функціонуванні онлайн-магазину.

Змодельований функціонал майбутньої веборієнтованої системи представлено у таблиці 1.2.

Таблиця 1.2. Змодельований функціонал веборієнтованої системи для електронної комерції

№	Назва функції	Опис функціоналу	Користувач
1	Реєстрація користувача	Надання можливості створення облікового запису з валідацією email	Покупець
2	Авторизація	Вхід у систему за логіном і паролем, зберігання токена сесії	Покупець, адміністратор
3	Перегляд каталогу товарів	Відображення списку товарів, фільтрація за категоріями, пошук	Усі
4	Перегляд сторінки товару	Перегляд детальної інформації про обраний товар	Усі
5	Додавання товару до кошика	Збереження товару у кошику, зміна кількості	Покупець
6	Оформлення замовлення	Вибір способу доставки й оплати, підтвердження даних	Покупець
7	Онлайн-оплата	Інтеграція з платіжним шлюзом (наприклад, LiqPay або Stripe)	Покупець
8	Перегляд історії замовлень	Перегляд попередніх покупок, їх статусів	Покупець
9	Додавання/редагування / видалення товару	Керування товарною базою через панель адміністратора	Адміністратор

10	Управління замовленнями	Перегляд, змінення статусу, підтвердження відправлення	Адміністратор
11	Керування користувачами	Перегляд зареєстрованих акаунтів, блокування за порушення	Адміністратор
12	Відправка сповіщень	Email-повідомлення про замовлення, підтвердження реєстрації	Система

На основі змодельованого функціоналу, було визначено варіанти використання системи основними користувачами та представлено в об'єктно-орієнтованому вигляді, а саме через діаграму варіантів використання (Use Case Diagram), яка є частиною методології UML.

Use Case Diagram дозволила відобразити взаємодію між зовнішніми акторами (користувачами системи) та функціональністю, яку їм надає система. Основними акторами системи було виділено:

1. Користувач (гість) - актор представляє неавторизованого відвідувача сайту. Його функціональні можливості обмежені лише переглядом інформації. Він взаємодіє з такими прецедентами:

- **«Переглянути каталог»** – перегляд списку доступних товарів, категорій;
- **«Переглянути товар»** – перегляд детальної інформації про окремий товар;
- **«Зареєструватися»** – створення облікового запису;
- **«Увійти в систему»** – автентифікація для переходу в статус зареєстрованого користувача.

2. Зареєстрований користувач (покупець) - після авторизації користувач набуває розширених прав і може виконувати ключові дії, пов’язані з електронною комерцією:

- **«Переглянути каталог»**
- **«Переглянути товар»**

- **«Додати до кошика»** – формування списку товарів для покупки;
- **«Оформити замовлення»** – підтвердження кошика, вибір способу доставки, контактних даних;
- **«Оплатити онлайн»** – ініціація та завершення платіжної транзакції через інтегровану платіжну систему;
- **«Отримати email-сповіщення»** – отримання повідомлень про стан замовлення, підтвердження реєстрації, відновлення пароля тощо.

3. Адміністратор - взаємодіє із системою через окремий адміністративний інтерфейс і має повні повноваження щодо управління ресурсами сайту:

- **«Керувати товарами»** – додавання, редагування або видалення товарів;
- **«Керувати замовленнями»** – перегляд, зміна статусу, обробка замовлень;
- **«Керувати користувачами»** – модерація користувачів, блокування, аналіз активності тощо;
- **«Отримати email-сповіщення»** – адміністративні повідомлення про критичні події (наприклад, нові замовлення, проблеми з оплатою).

4. Платіжна система - зовнішній актор, що відповідає за здійснення фінансових транзакцій:

- **«Оплатити онлайн»** – виконує обробку платіжних запитів, перевірку валідності картки, підтвердження переказу коштів.

5. Поштовий сервіс - інтегрований сервіс, який відповідає за обмін повідомленнями та деякі технічні функції:

- **«Отримати email-сповіщення»** – надсилає користувачу підтвердження реєстрації, інформацію про замовлення, повідомлення про зміни;
- також може бути залучений до процесів **автентифікації**, **відновлення пароля** та **підтвердження email-адреси**.

Змодельована діаграма використання для системи електронної комерції представлена на рисунку 2.1.

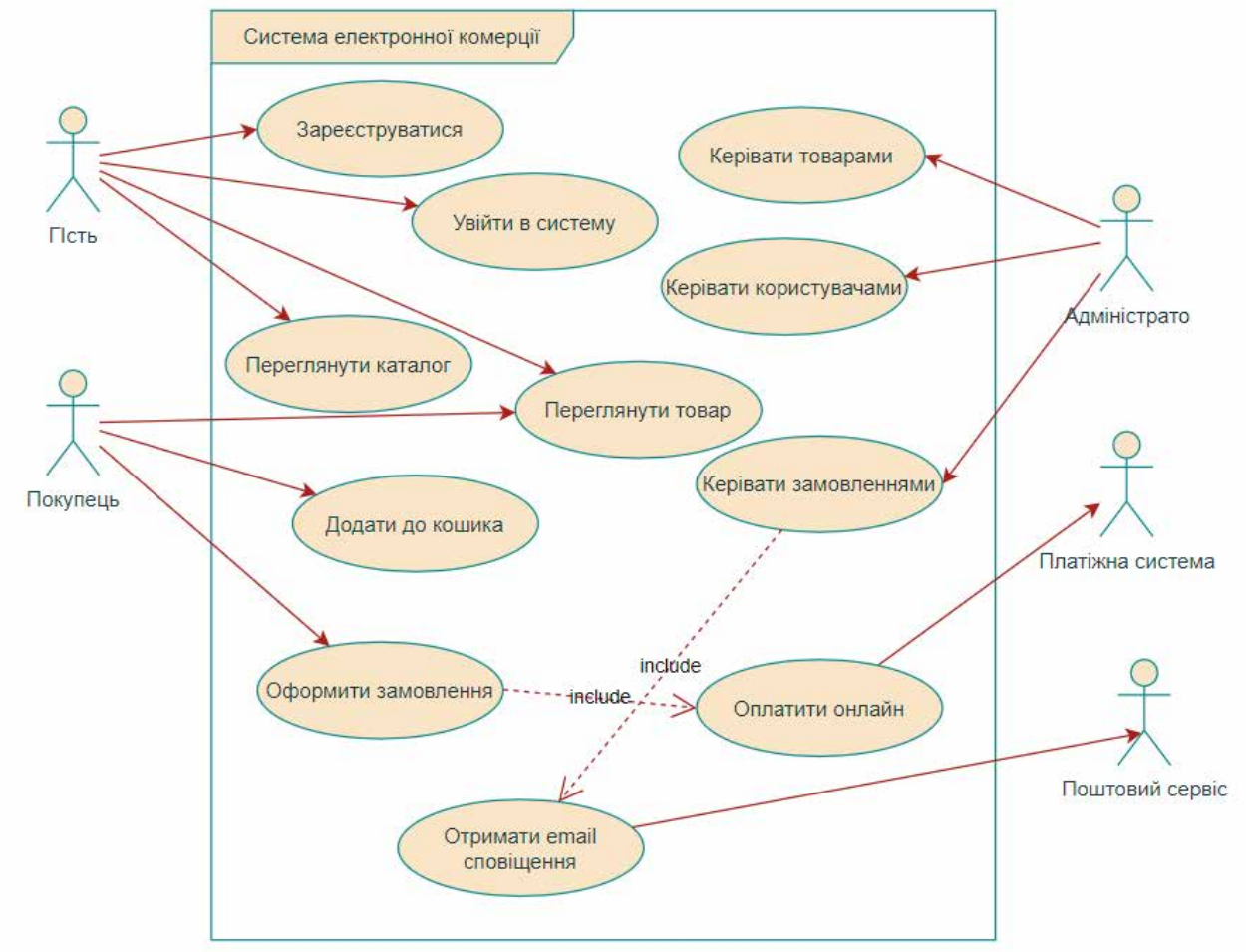

Рис. 2.1. Use Case модель системи електронної комерції

У цьому розділі було здійснено моделювання основних варіантів використання веборієнтованої системи електронної комерції на основі визначених ролей користувачів. Виділено ключових акторів – гостя, зареєстрованого користувача, адміністратора, платіжну систему та поштовий сервіс – і встановлено їхню взаємодію із системою через відповідні прецеденти. Побудова діаграми варіантів використання дозволила наочно представити функціональні можливості системи та закласти основу для подальшого етапу проектування – деталізації сценаріїв і розробки архітектури програмного забезпечення.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель даних є ключовим етапом проєктування бази даних системи електронної комерції, оскільки забезпечує структуроване уявлення про інформацію, яка зберігається та обробляється системою, незалежно від конкретної реалізації СКБД. У межах даної моделі було виділено основні сутності, їх атрибути, а також зв’язки між ними, виходячи з функціональних потреб платформи.

Основні сутності системи виступатиме користувач, товар, категорія, замовлення, деатлі замовлення, кошик, позиції кошика та платіж.

Сутність «Користувач» є базовою складовою системи, яка зберігає дані про всіх осіб, що мають доступ до функціоналу платформи. Це можуть бути як звичайні покупці, так і адміністратори системи. Кожен користувач має унікальний ідентифікатор user_id, а також електронну пошту (email), яка виступає логіном для авторизації. Для захисту персональних даних пароль зберігається у вигляді хешу (password_hash). Крім того, зберігається ім’я користувача (name), роль (role), яка визначає рівень доступу до функцій (наприклад, “customer” або “admin”), дата реєстрації (registration_date) та статус (status), який може свідчити про активність облікового запису (наприклад, “активний”, “заблокований”). Один користувач може мати лише один активний кошик, а також бути автором кількох замовлень.

Сутність «Товар» описує усі позиції, доступні для перегляду та покупки в електронному каталозі. Кожен товар має унікальний ідентифікатор product_id, а також назву (name), опис (description), актуальну ціну (price) та кількість на складі (stock_quantity). Для організації та фільтрації товари належать до певних категорій (category_id). Також передбачено збереження зображення товару за посиланням (image_url) і дату додавання до каталогу (created_at). Один товар може бути представлений у кількох замовленнях та входити до складу кількох кошиків. Така структура дозволяє динамічно оновлювати каталог і підтримувати актуальність пропозицій.

Сутність «Категорія» призначена для класифікації товарів за типами, що дозволяє формувати зручну навігацію у вигляді ієрархічної структури. Кожна категорія має унікальний ідентифікатор category_id, назву (name) і може мати посилання на батьківську категорію (parent_id), утворюючи дерево категорій та підкатегорій. Наприклад, категорія "Побутова техніка" може містити підкатегорії "Холодильники", "Пральні машини" тощо. Такий підхід забезпечує гнучке групування товарів і полегшує користувачам пошук потрібної продукції.

Сутність «Замовлення» фіксує факт покупки користувачем одного чи кількох товарів. Кожне замовлення ідентифікується унікальним order_id і містить посилання на користувача (user_id), який його оформив. Додатково зберігається дата оформлення (order_date), загальна сума замовлення (total_amount), його статус (status – наприклад, “нове”, “у виконанні”, “доставлено”), адреса доставки (delivery_address) та статус оплати (payment_status). Замовлення є ключовою сутністю для аналітики, обліку та управління логістикою. Воно зв’язане з деталями замовлення (OrderItem) і платежами.

Сутність «Деталі замовлення» відображає конкретні позиції товарів, що входять до складу певного замовлення. Вона містить унікальний ідентифікатор order_item_id, а також зовнішні ключі до замовлення (order_id) та товару (product_id). Крім того, вказується кількість одиниць товару (quantity) та ціна на момент покупки (price_at_purchase). Ця структура дозволяє точно зберігати історію замовлення навіть у разі зміни ціни товару після покупки. Таким чином, замовлення може включати кілька записів у таблиці OrderItem, кожен з яких описує окремий товар у конкретній кількості.

Сутність «Кошик» забезпечує можливість зберігання обраних товарів користувача до моменту оформлення замовлення. Вона містить унікальний ідентифікатор cart_id, зовнішній ключ до користувача (user_id), якому належить кошик, а також дату створення (created_at). Кошик має тимчасовий характер, він оновлюється в реальному часі залежно від дій користувача: додавання або видалення товарів, зміни кількості тощо. Оформлення замовлення призводить до трансформації вмісту кошика у замовлення та очищення його після завершення процесу.

Сутність «Позиції кошика» зберігає дані про кожен конкретний товар, доданий користувачем до кошика. Вона включає унікальний ідентифікатор cart_item_id, зовнішні ключі до кошика (cart_id) та товару (product_id), а також кількість одиниць (quantity). Це дозволяє користувачеві зберігати декілька товарів одночасно з індивідуальною кількістю кожного. Позиції кошика динамічно оновлюються та забезпечують зручний механізм формування майбутнього замовлення.

Сутність «Платіж» відповідає за збереження інформації про фінансові транзакції, здійснені користувачами після оформлення замовлення. Вона має унікальний ідентифікатор payment_id, посилання на відповідне замовлення (order_id), дату платежу (payment_date), суму (amount), метод оплати (payment_method, наприклад, картка, Google Pay, банківський переказ) та статус платежу (payment_status, наприклад, “оплачено”, “очікується”, “відхилено”). Дана сутність є важливою для звітності, обліку та інтеграції з платіжними шлюзами.

Зв’язки між сутностями:

- Один користувач може мати багато замовлень (User 1–∞ Order).
- Одне замовлення може містити багато товарів через OrderItem.
- Один товар може належати до однієї категорії (Product ∞–1 Category).
- Один кошик належить одному користувачу, але може містити багато позицій (Cart 1–∞ CartItem).
- Один платіж прив’язаний до одного замовлення (Order 1–1 Payment).

Також варто врахувати, що усі поля, що містять паролі, передбачають використання шифрування (наприклад, bcrypt), зв’язки між сутностями реалізуються через зовнішні ключі та підтримують референційну цілісність, передбачено логування транзакцій, помилок і активності користувачів (у розширеній версії моделі).

Модель даних веборієнтованої системи електронної комерції представлена на рисунку 2.2.

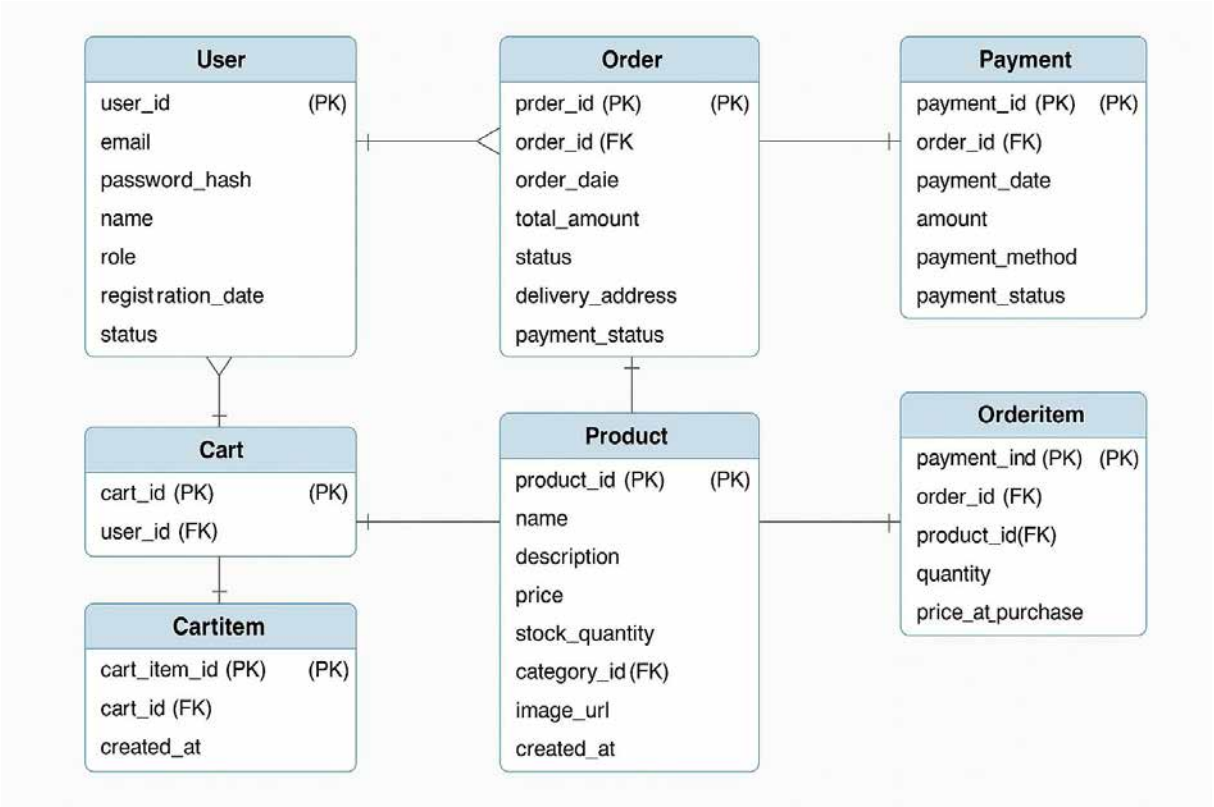

Рис. 2.2. Модель даних системи електронної комерції

Побудована логічна модель даних формує основу для подальшої реалізації системи електронної комерції. Було визначено ключові сутності, їхні атрибути та взаємозв’язки, що забезпечують цілісне представлення інформаційної структури системи. Розроблена модель відображає основні бізнес-процеси: управління користувачами, обробку замовлень, облік товарів, ведення кошика та інтеграцію з платіжною й поштовою інфраструктурою.

ER-діаграма логічної моделі даних забезпечує наочне уявлення про структуру бази даних і слугує проміжним етапом між концептуальним проєктуванням і фізичною реалізацією. Отримані результати сприяють ефективному зберіганню, обробці та доступу до даних у системі, що є критично важливим для стабільної роботи веборієнтованої платформи електронної торгівлі.

2.2 Вибір системи управління інформаційною базою

Розробка надійної, масштабованої та швидкодіючої веборієнтованої системи електронної комерції потребує ретельного вибору системи управління базами даних (СУБД), яка буде відповідати функціональним і нефункціональним вимогам до проєкту.

Основні критерії вибору СУБД:

- Продуктивність і швидкодія – важливий фактор для обробки великої кількості одночасних запитів від користувачів, особливо під час пікових навантажень.
- Масштабованість – можливість безперешкодного розширення бази даних зі зростанням кількості товарів, замовлень і користувачів.
- Сумісність з вебтехнологіями – підтримка сучасних фреймворків та мов програмування (PHP, Python, Node.js, тощо).
- Надійність та забезпечення цілісності даних – підтримка транзакцій, механізмів резервного копіювання та захисту даних.
- Ліцензійна політика та вартість – відкритість, доступність, умови ліцензування.
- Спільнота та підтримка – наявність документації, активного співтовариства, регулярних оновлень.

З урахуванням вищезазначених критеріїв, для реалізації даної системи було обрано PostgreSQL – потужну об’єктно-реляційну систему управління базами даних з відкритим кодом.

Основні переваги PostgreSQL у контексті електронної комерції:

- Висока надійність та відповідність стандартам ACID – забезпечує цілісність транзакцій при оплаті замовлень, зміні статусів, тощо.
- Гнучка робота з типами даних – можливість зберігати як структуровані, так і напівструктуровані дані (наприклад, JSON).
- Масштабованість і оптимізація запитів – завдяки підтримці індексів, тригерів, stored procedures тощо.

- Широка інтеграція з популярними фреймворками – підтримка Django, Laravel, Node.js, Spring та ін.
- Безкоштовна ліцензія (OSI-Approved PostgreSQL License) – зменшення загальної вартості розробки.
- Активна спільнота та велика база знань – пришвидшує розробку та усунення потенційних помилок.

Альтернативи, що розглядалися:

MySQL / MariaDB – прості у налаштуванні, але поступаються PostgreSQL за складністю запитів, роботою з транзакціями та типами даних.

MongoDB – підходить для гнучких NoSQL-структур, однак менш оптимальна для класичних табличних реляцій між товарами, користувачами та замовленнями.

SQLite – надто легка для проєктів із багатьма одночасними запитами та високим навантаженням.

Таким чином, СУБД PostgreSQL була обрана як найбільш придатна для потреб веборієнтованої системи електронної комерції завдяки своїй масштабованості, функціональності, підтримці транзакцій та відкритості. Її використання забезпечує стабільність, безпеку та гнучкість у роботі з даними в умовах комерційного середовища.

2.3 Створення інформаційної бази

Після побудови логічної моделі даних та вибору відповідної системи управління базами даних було здійснено етап створення інформаційної бази, яка забезпечує зберігання, доступ і обробку даних у веборієнтованій системі електронної комерції. Для реалізації бази даних використано СУБД PostgreSQL, яка забезпечує гнучку роботу з реляційною структурою та відповідає усім вимогам до надійності, продуктивності та масштабованості.

Основні етапи створення інформаційної бази:

1. Ініціалізація бази даних - було створено окрему базу даних під назвою ecommerce_system, із заздалегідь визначеними схемами, таблицями та індексами.

2. Реалізація таблиць - на основі логічної моделі створено SQL-скрипти, які описують структуру таблиць (рис. 2.3).

```sql
CREATE TABLE Users (
  user_id SERIAL PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  name VARCHAR(100),
  role VARCHAR(50) DEFAULT 'buyer',
  registration_date TIMESTAMP DEFAULT NOW(),
  status BOOLEAN DEFAULT TRUE
);

CREATE TABLE Products (
  product_id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  price NUMERIC(10,2) NOT NULL,
  stock_quantity INTEGER,
  category_id INTEGER REFERENCES Categories(category_id),
  image_url TEXT,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Рис. 2.3. Приклад опису структури таблиць на прикладі таблиці користувачів

Аналогічно створюються таблиці для Orders, OrderItems, Cart, CartItems, Payments, Categories тощо.

3. Забезпечення цілісності даних - було реалізовано зовнішні ключі, унікальні обмеження, перевірки на null, а також логічні залежності між сутностями для забезпечення референційної цілісності.

4. Індексація та оптимізація - для підвищення швидкодії системи при виконанні частих запитів створено індекси по полях email, product_id, order_id, що пришвидшує пошук і фільтрацію даних.

5. Заповнення початковими даними (seed data) - для тестування основного функціоналу було створено тестові записи користувачів, товарів та замовлень. Це дало змогу перевірити роботу CRUD-операцій і коректність взаємодії між таблицями.

6. Резервне копіювання - налаштовано механізм регулярного бекапу бази даних з використанням утиліти pg_dump, що дозволяє зберігати поточний стан даних у разі збоїв.

У результаті даного етапу було створено повноцінну інформаційну базу, яка забезпечує надійне зберігання структурованих даних для веборієнтованої системи електронної комерції. Вона є основою для взаємодії між клієнтською частиною, адміністративним модулем та зовнішніми сервісами (платіжними й поштовими системами). Завдяки правильно спроєктованій структурі таблиць і оптимізації запитів база даних демонструє високу продуктивність та готовність до масштабування.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення описує логічне розбиття системи на окремі модулі та компоненти, що виконують специфічні функції, а також принципи їх взаємодії. Така структура сприяє ефективному управлінню проєктом, спрощує розробку, тестування, масштабування та подальший супровід системи.

У процесі розробки було обрано багаторівневу архітектуру (multi-tier architecture), яка передбачає поділ системи на три основні рівні: клієнтський, серверний та рівень даних.

Клієнтський рівень (Frontend) - реалізований з використанням HTML, CSS, JavaScript та фреймворку React.js. Забезпечує інтерфейс взаємодії з користувачем, обробку подій та формування запитів до сервера.

Серверний рівень (Backend) - розроблений за допомогою мови програмування Node.js у поєднанні з фреймворком Express. Відповідає за обробку логіки запитів, авторизацію, валідацію, роботу з базою даних, а також взаємодію з зовнішніми сервісами (оплата, email).

Рівень даних (Database Layer) - представлений СУБД PostgreSQL. Зберігає інформацію про користувачів, товари, замовлення, транзакції тощо. Взаємодіє з серверним рівнем через SQL-запити та ORM (Object-Relational Mapping).

Основними програмними модулями майбутньої веборієнтованої системи електронної комерції виступитимуть:

AuthModule – автентифікація та авторизація користувачів, керування сесіями.

UserModule – профіль користувача, історія замовлень, редагування особистих даних.

ProductModule – відображення каталогу, пошук, фільтрація, детальна інформація про товар.

CartModule – зберігання та редагування кошика користувача.

OrderModule – оформлення замовлень, облік статусу, перегляд замовлень.

PaymentModule – інтеграція з платіжною системою, обробка платежів.

AdminModule – керування товарами, користувачами, замовленнями (доступ лише для адміністратора).

NotificationModule – надсилання email-сповіщень за допомогою SMTP або сторонніх сервісів (наприклад, SendGrid).

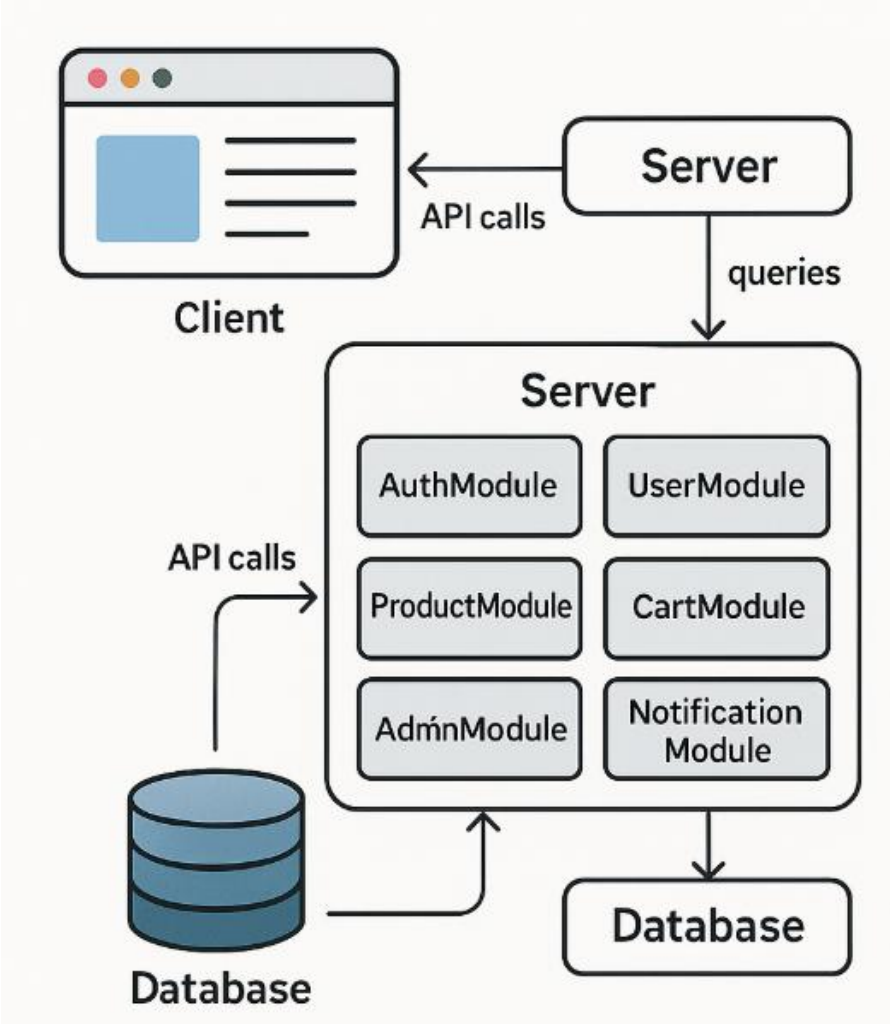

Рис. 3.1. Взаємодія програмних модулів майбутньої веборієнтованої системи електронної комерції.

Всі модулі взаємодіють через API-запити у форматі REST. Кожен запит передається від клієнта до сервера, де обробляється відповідним модулем, після чого сервер повертає відповідь у форматі JSON.

Організаційна структура програмного забезпечення побудована з дотриманням принципів модульності, розподілу відповідальності та розширюваності. Такий підхід забезпечує простоту підтримки системи, дозволяє гнучко інтегрувати нові функції та адаптувати продукт до змін бізнес-вимог. Обрані технології та архітектура відповідають сучасним стандартам веброзробки та гарантують стабільну роботу платформи електронної комерції.

3.2 Вибір інструментарію для розробки

Процес створення веборієнтованої системи електронної комерції вимагає ретельного підходу до вибору інструментарію. Від правильності цього вибору залежить ефективність розробки, стабільність функціонування системи, зручність супроводу та її масштабованість у майбутньому.

Основною мовою програмування для реалізації клієнтської та серверної частин проєкту було обрано JavaScript (ES6+). Ця мова є універсальною у веброзробці, забезпечує високий рівень гнучкості, має активну спільноту та підтримується більшістю сучасних фреймворків. Для роботи з базою даних використовувалась мова SQL, що дала змогу створити запити до реляційної СУБД PostgreSQL з метою зберігання та обробки інформації про користувачів, товари, замовлення та транзакції.

Для створення клієнтської частини інтерфейсу користувача було обрано бібліотеку React.js. Вона забезпечує компонентну структуру, що дає змогу багаторазово використовувати елементи інтерфейсу, а також оптимізує відображення сторінок завдяки віртуальному DOM. На серверній частині застосовано середовище Node.js разом із фреймворком Express.js, які відповідають за обробку HTTP-запитів, реалізацію логіки взаємодії з клієнтом, авторизацію та перевірку даних. Для взаємодії з базою даних використано

бібліотеку Sequelize – ORM-рішення, що дає змогу працювати з PostgreSQL у вигляді об’єктів і абстрагуватися від написання складних SQL-запитів.

Стилізація інтерфейсу здійснювалась за допомогою бібліотек Bootstrap і Tailwind CSS, які прискорюють розробку адаптивного та естетично привабливого дизайну. Усі ці засоби у поєднанні створюють гнучке і масштабоване середовище для реалізації функціоналу електронної торгівлі.

Серцем інформаційної бази стала система керування базами даних PostgreSQL, яка є потужною об’єктно-реляційною СУБД. Вона підтримує транзакційність, забезпечує високу продуктивність, має вбудовані механізми забезпечення цілісності даних, а також добре інтегрується з іншими частинами проєкту.

Розробка системи здійснювалась у середовищі Visual Studio Code – легкому, проте функціональному редакторі коду з численними розширеннями для JavaScript, React, SQL, REST API та ін. Для тестування API-запитів використовувалися утиліти Postman та Insomnia, а для роботи з базою даних – графічна оболонка pgAdmin, яка забезпечує візуалізацію структури таблиць, написання SQL-запитів і керування даними.

Контроль версій реалізовано за допомогою системи Git, а хмарне зберігання та командна робота організовані через платформу GitHub. Такий підхід дає змогу ефективно відстежувати зміни у коді, створювати гілки, повертатися до стабільних версій, а також організовувати розробку в команді.

У межах реалізації платіжного функціоналу було розглянуто сервіси Stripe і LiqPay як можливі рішення для онлайн-оплати. Для обробки електронних сповіщень (наприклад, підтвердження замовлень чи відновлення пароля) використовувалися сервіси Nodemailer і SendGrid. Задля зберігання конфіденційних налаштувань, таких як ключі API або паролі до поштових сервісів, застосовано бібліотеку dotenv, що дозволяє безпечно керувати змінними середовища.

Рис. 3.2. Технологічний стек проекту розробки

Таким чином, обраний стек технологій відповідає сучасним стандартам веброзробки та дозволяє реалізувати стабільну, масштабовану та функціональну систему електронної комерції. Використання відкритих, гнучких і добре підтримуваних рішень також сприяє зниженню вартості проєкту, пришвидшенню розробки та забезпечує простоту у подальшому супроводі програмного продукту.

3.3 Програмування програмних модулів

Етап програмування є центральним у процесі реалізації веборієнтованої системи електронної комерції, оскільки саме в ньому здійснюється втілення попередньо спроєктованої логіки та функціоналу в вигляді працюючого програмного коду. Усі функціональні частини системи були реалізовані у вигляді окремих програмних модулів, що відповідають за певні бізнес-процеси та взаємодіють між собою через API-запити.

Програмування розпочалося з побудови серверної частини. Основу бекенду склала платформа Node.js у поєднанні з фреймворком Express.js. Було створено маршрути для обробки запитів від клієнтської частини, зокрема запитів на реєстрацію, авторизацію, перегляд товарів, створення замовлень, проведення оплати та отримання підтверджень. Усі маршрути пов’язані з відповідними контролерами, які реалізують логіку обробки даних, перевірку прав доступу, валідацію введених користувачем даних та взаємодію з базою даних через ORM Sequelize.

Модуль автентифікації (AuthModule) забезпечує реєстрацію нових користувачів, вхід до системи та захист маршрутів за допомогою JWT-токенів. Було реалізовано механізм шифрування паролів з використанням алгоритму bcrypt, що гарантує безпеку персональних даних користувачів. У свою чергу, модуль управління користувачами (UserModule) дозволяє переглядати особистий профіль, змінювати облікову інформацію та переглядати історію попередніх замовлень.

Модуль управління товарами (ProductModule) відповідає за виведення каталогу, пошук за назвою або категорією, перегляд опису товарів та доступ до зображень. Усі запити до товарів оптимізовано через індексацію полів у базі даних, що забезпечує високу швидкодію пошуку.

Кошик (CartModule) реалізовано як тимчасове сховище товарів, обране користувачем перед покупкою. Користувач має змогу додавати або видаляти позиції, а також оновлювати їх кількість. Дані з кошика передаються в модуль замовлень (OrderModule) для створення остаточного замовлення. Тут також реалізовано обробку статусів замовлення – від моменту створення до підтвердження оплати.

Модуль оплати (PaymentModule) відповідає за інтеграцію з платіжною системою. Було реалізовано тестову взаємодію з сервісом Stripe або LiqPay, включаючи обробку зворотного з'єднання (callback) після підтвердження платежу. Результати оплати зберігаються у відповідній таблиці бази даних, що дозволяє відслідковувати історію транзакцій.

Окрему увагу приділено модулю сповіщень (NotificationModule), який реалізує надсилання електронних листів користувачам. За допомогою бібліотеки Nodemailer було налаштовано SMTP-з'єднання з поштовим сервером для надсилання листів підтвердження, нагадувань та інформації про статус замовлення.

На клієнтському рівні програмування здійснювалося з використанням React.js. Для кожної сторінки інтерфейсу було створено окремі компоненти: головна сторінка, каталог товарів, сторінка товару, кошик, форма оформлення замовлення, кабінет користувача, тощо. Взаємодія з бекендом відбувається через HTTP-запити за допомогою бібліотеки Axios. Всі компоненти мають адаптивний дизайн і протестовані на різних розмірах екранів для забезпечення коректної роботи на мобільних пристроях.

У результаті програмування було реалізовано повний набір функціональності системи електронної комерції: від реєстрації користувача до фінальної оплати й обробки замовлення адміністратором. Всі модулі побудовано з урахуванням принципів повторного використання коду, масштабованості та безпеки.

4 ТЕСТУВАННЯ СИСТЕМИ ТА ВИМОГИ

4.1 Методологія тестування

Тестування веборієнтованої системи електронної комерції здійснювалося на всіх ключових рівнях архітектури застосунку відповідно до обраного технологічного стеку: JavaScript, React, Node.js, Express, PostgreSQL, Git та середовище Visual Studio Code. Основною метою було забезпечення надійності, коректності бізнес-логіки, зручності інтерфейсу та стабільності API.

Для перевірки клієнтської частини системи, реалізованої з використанням React, проводилось модульне тестування компонентів. Основну увагу було приділено перевірці коректної візуалізації елементів інтерфейсу, передачі пропсів, обробці подій (наприклад, додавання товару в кошик) та валідації форм. Для реалізації тестів використовувалися інструменти Jest та React Testing Library, які дозволяють створювати симуляції взаємодій з компонентами у віртуальному DOM-середовищі.

Серверна логіка застосунку, побудована на Node.js із використанням фреймворку Express, була протестована шляхом модульного й інтеграційного тестування. Тестування API включало перевірку маршрутів, обробку запитів, автентифікацію, взаємодію з базою даних і обробку помилок. Для цього використовувалися утиліти Postman та Insomnia, які дозволяють тестувати різні сценарії взаємодії з сервером, у тому числі авторизовані запити та обробку граничних випадків (наприклад, недостатній баланс товару, помилкові дані користувача тощо).

Крім ручного тестування REST API, було реалізовано часткову автоматизацію за допомогою бібліотеки Mocha/Chai, яка дала змогу проводити серійні перевірки бізнес-логіки та коректності відповіді сервера у тестовому середовищі.

На рівні бази даних особлива увага приділялась перевірці відповідності структури таблиць проєктній моделі (ER-діаграмі), коректності зв’язків між

сутностями, налаштуванню зовнішніх ключів та індексів. Для візуального аналізу і тестування SQL-запитів використовувалась графічна оболонка pgAdmin, яка дозволила виявляти потенційні помилки у запитах, перевіряти логіку тригерів, обмежень та актуальність даних у тестовому середовищі.

Окремо було протестовано сценарії транзакцій при оформленні замовлень та здійсненні платежів, щоб гарантувати цілісність даних навіть у разі збоїв.

Під час розробки активно використовувалась система контролю версій Git. Усі зміни у кодовій базі фіксувались у репозиторії на GitHub, що дозволяло швидко відстежувати внесені правки, повертатись до стабільних версій і проводити колективний код-рев'ю. Завдяки використанню гілок (feature branches) було забезпечено ізоляцію нових функцій від основної версії застосунку до моменту їх повної перевірки.

У межах тестування системи враховувались також питання безпеки. Для зберігання конфіденційних налаштувань, таких як ключі API, токени авторизації або SMTP-параметри, використовувалась бібліотека dotenv, що дозволила уникнути публічного зберігання чутливої інформації в коді.

4.2 Проведення тестування

Процес тестування веборієнтованої системи електронної комерції охоплював усі ключові компоненти архітектури: клієнтську частину (frontend), серверну логіку (backend), базу даних, а також інтеграцію сторонніх сервісів. Тестування виконувалося ітераційно протягом усіх етапів розробки з метою оперативного виявлення та усунення помилок, а також забезпечення стабільності системи при розширенні функціоналу.

Тестування інтерфейсу користувача (UI)

Під час тестування клієнтської частини особлива увага приділялася зручності та передбачуваності взаємодії користувача з інтерфейсом. Перевірялися основні сценарії використання системи, зокрема:

- реєстрація та авторизація користувача;
- додавання товарів до кошика;

- перегляд списку товарів і фільтрація за категоріями;
- оформлення замовлення;
- підтвердження платежу;
- отримання електронного листа з підтвердженням.

Було проведено ручне функціональне тестування UI-компонентів у браузерах **Google Chrome** та **Mozilla Firefox**, що дозволило оцінити відображення інтерфейсу в різних середовищах. Виявлені візуальні недоліки (наприклад, неправильне масштабування або некоректне відображення повідомлень про помилки) були усунені шляхом корекції CSS-стилів і логіки рендерингу компонентів.

Тестування REST API

Для перевірки серверної частини застосунку активно використовувались **Postman** і **Insomnia**. Було складено набір колекцій з API-запитами, які охоплювали всі основні маршрути: реєстрація, логін, CRUD-операції з товарами, категоріями, замовленнями та платежами. Для кожного запиту перевірялися:

- коректність HTTP-статусів (200, 201, 400, 401, 404, 500 тощо);
- вміст тіла відповіді (response body);
- валідація введених даних;
- обробка помилок у разі некоректних запитів або відсутності авторизації.

Крім позитивних сценаріїв, тестувалися негативні випадки, зокрема спроби неавторизованого доступу до закритих маршрутів, передача некоректних форматів даних та робота API при відсутності зв’язку з базою даних.

Перевірка роботи бази даних

Тестування бази даних PostgreSQL проводилось як вручну (через **pgAdmin**), так і автоматично у процесі виконання API-запитів. Усі таблиці були перевірені на відповідність проєктній структурі. Було протестовано:

- коректність створення записів (INSERT);
- оновлення даних (UPDATE);
- зв’язки між таблицями (JOIN, FOREIGN KEYS);

- обмеження (NOT NULL, UNIQUE, CHECK);

- обробку транзакцій, зокрема при оформленні замовлень та платежах.

Тестування дозволило виявити й усунути кілька структурних недоліків, наприклад, дублювання товарів через відсутність індексу або затримки у виконанні запитів при складних JOIN-операціях.

Перевірка платіжного функціоналу

Особливу увагу приділено тестуванню платіжного модуля. Було налаштовано тестові ключі для сервісів **Stripe** і **LiqPay**, які дозволяють проводити симуляцію транзакцій без фактичного списання коштів. Перевірялися:

- генерація сесії оплати;
- переадресація на платіжну сторінку;
- повернення користувача на сайт після оплати;
- збереження статусу транзакції в базі даних.

Під час тестування було виявлено необхідність обробки сценаріїв невдалого платежу, зокрема таких випадків, як відміна операції користувачем, помилка при введенні платіжних даних або відхилення транзакції банком. Ці ситуації виявились критично важливими для забезпечення стабільної роботи системи та підвищення зручності користування. Відповідно, у коді були впроваджені механізми обробки помилок, що дозволяють коректно реагувати на подібні інциденти: надається зрозуміле повідомлення користувачу, записуються логи для подальшого аналізу, а також реалізовані можливості повторного здійснення платежу або повернення до попереднього етапу. Завдяки цьому система стала більш надійною та дружньою до кінцевого користувача.

Перевірка відправлення електронних листів

Було протестовано функціонал відправлення листів на електронну пошту користувача з підтвердженням реєстрації, відновленням пароля та інформацією

про замовлення. Для цього застосовувались **Nodemailer** (у поєднанні зі SMTP-сервером) та **SendGrid**. Було перевірено:

- доставку листів на поштові скриньки Gmail, Outlook;
- коректність шаблонів HTML;
- швидкість відправки;
- обробку помилок при недоступності сервісу.

Приклад тест-кейсів функціонального тестування представлено у таблиці 4.1.

Таблиця 4.1. Приклади тест-кейсів функціонального тестування

№	**Назва функції**	**Опис тест-кейсу**	**Вхідні дані**	**Очікуваний результат**	**Статус**
1	Реєстрація користувача	Перевірка створення нового облікового запису	Ім’я, email, пароль	Користувача створено, повертається токен	✓
2	Авторизація	Вхід до системи з правильними обліковими даними	Email, пароль	Успішний вхід, редирект на головну сторінку	✓
3	Невдала авторизація	Вхід до системи з неправильним паролем	Email, неправильний пароль	Повідомлення про помилку входу	✓
4	Додавання товару до кошика	Користувач натискає кнопку	Обраний товар	Товар з’являється в	✓

		“Додати в кошик”		кошику з кількістю 1	
5	Оформлення замовлення	Оформлення покупки через форму замовлення	Адреса, метод оплати, кошик	Замовлення створено, повідомлення підтвердження	✓
6	Обробка платежу через Stripe	Симуляція успішної оплати	Дані тестової картки Stripe	Отримано статус “оплачено”, замовлення оновлено	✓
7	Відправка електронного підтвердження	Перевірка надсилання листа після оформлення замовлення	Email користувача	Лист надіслано, лог у консолі / повідомлення в UI	✓
8	Захист маршрутів API	Перевірка доступу до захищених ресурсів без токена	API-запит без авторизації	Відповідь 401: доступ заборонено	✓
9	Перевірка обмежень БД	Додавання користувача з email, що вже існує	Повторний email	Відмова у збереженні, повідомлення про дублювання	✓

10	Перегляд товарів за категоріями	Перевірка фільтрації товарів на сторінці каталогу	Обрана категорія	Відображаються лише товари з відповідною категорією	✓

В результаті тестування система показала стабільну роботу основних функцій, швидкий час відповіді сервера, та високу надійність при обробці транзакцій і взаємодії з користувачем. Усі виявлені помилки та баги були задокументовані у репозиторії GitHub у вигляді **Issue-записів** та виправлені у наступних комітах. Після фінального тестування проєкт був підготовлений до розгортання у production-середовищі.

4.3 Вимоги та рекомендації щодо впровадження

Впровадження веборієнтованої системи електронної комерції потребує дотримання ряду технічних, організаційних та безпекових вимог для забезпечення її стабільної роботи, масштабованості та зручності у користуванні як для покупців, так і для адміністраторів.

Система базується на стеку технологій JavaScript, Node.js, React, Express та PostgreSQL, що робить її сумісною з більшістю сучасних хостинг-платформ, включаючи Heroku, Render, Vercel, AWS та інші. Для коректної роботи серверної частини необхідно забезпечити підтримку середовища виконання Node.js (версія не нижче 16.0), а також встановлену систему керування базами даних PostgreSQL. Система повинна розгортатися у середовищі, яке дозволяє керувати змінними оточення (наприклад, .env), забезпечуючи конфіденційність API-ключів, SMTP-доступу та параметрів безпеки.

Також доцільно мати налаштоване резервне копіювання бази даних, логування подій, контроль навантаження та захист від DDoS-атак, особливо у періоди сезонного підвищення активності (розпродажі, свята тощо).

Рекомендується використовувати HTTPS-з'єднання та інтеграцію з Cloudflare або Let’s Encrypt для шифрування трафіку.

Перед впровадженням у реальне середовище слід завершити повноцінне тестування усіх функціональних модулів: реєстрація, автентифікація, каталог товарів, фільтрація, кошик, замовлення, платіжна інтеграція, адміністрування контенту. У системі має бути реалізована можливість гнучкого налаштування ролей (користувач, адміністратор) з відповідними правами доступу до певного функціоналу. Крім того, рекомендується впровадити систему електронної пошти для підтверджень замовлень, відновлення паролів та комунікації з клієнтами.

Для забезпечення якісного користувацького досвіду (UX) слід звернути увагу на адаптивність інтерфейсу для мобільних пристроїв, логічну структуру навігації, швидкість завантаження сторінок та інтуїтивно зрозумілий процес оформлення замовлення. Необхідно передбачити обробку всіх типових помилок – від невірного логіну до нестачі товару на складі – з відповідними повідомленнями для користувача.

З точки зору організації впровадження, важливо визначити відповідальну особу або команду, яка здійснюватиме технічну підтримку, оновлення контенту, моніторинг замовлень та комунікацію з клієнтами. Рекомендується створити адміністративну панель з можливістю керування товарами, категоріями, замовленнями та користувачами. Це дозволить не лише автоматизувати основні бізнес-процеси, а й зменшити потребу у сторонніх ресурсах.

Важливо також підготувати інструкції з користування системою, як для персоналу, так і для кінцевих користувачів. У випадку інтеграції з платіжними сервісами (наприклад, Stripe або LiqPay), слід укласти відповідні договори, перевірити юридичні аспекти обробки платежів та дотримання вимог законодавства, включно з GDPR (у разі обробки персональних даних громадян ЄС).

Однією з ключових вимог до впровадження є забезпечення інформаційної безпеки. Паролі мають зберігатися виключно у вигляді хешів (bcrypt), усі дані, що передаються мережею, повинні бути зашифровані (TLS/SSL), а важливі

маршрути API – захищені автентифікацією з токенами доступу (JWT або OAuth). Регулярне оновлення залежностей, контроль прав доступу та аудит логів є необхідною умовою для уникнення вразливостей.

Впровадження комплексної системи моніторингу та аналітики є критично важливим етапом для будь-якого цифрового продукту. По-перше, моніторинг технічних помилок за допомогою таких інструментів, як LogRocket, Sentry або Prometheus, дозволяє оперативно виявляти та усувати збої в роботі системи. Це не тільки запобігає негативному досвіду користувачів, але й мінімізує потенційні фінансові втрати, що можуть бути спричинені некоректною роботою сервісу. Зокрема, LogRocket дозволяє відтворювати сесії користувачів, бачачи, що саме вони робили та які помилки виникали, тоді як Sentry фокусується на зборі та агрегації інформації про помилки з різних частин системи. Prometheus, у свою чергу, є потужним інструментом для збору метрик і моніторингу продуктивності інфраструктури в реальному часі.

Мінімальні системні вимоги до розробленої системи прдеставлені у таблиці 4.2.

Таблиця 4.2. Мінімальні системні вимоги для розгортання вебзастосунку

Компонент	Мінімальна конфігурація
Операційна система	Ubuntu Server 20.04+ / Windows Server 2019 / macOS Monterey
Процесор (CPU)	2 ядра, 2.0 GHz або вище
Оперативна пам’ять	Від 2 ГБ (рекомендовано 4 ГБ для стабільної роботи)
Дисковий простір	Від 10 ГБ (залежно від кількості медіафайлів та обсягу даних)
Node.js	Версія 16.x або вище
PostgreSQL	Версія 13.x або вище

Вебсервер (опційно)	Nginx або Apache для зворотного проксіювання та SSL
Зовнішній доступ	Порт 80/443 відкритий, налаштовані правила файрволу
Додаткове ПЗ	Git, dotenv, PM2 або systemd для процес-менеджменту
Середовище виконання	Підтримка `.env`, cron, планувальника завдань (у разі автоматичних дій)

Впровадження будь-якої нової системи, чи то програмного забезпечення, чи комплексної організаційної зміни, завжди супроводжується певними ризиками. Ці ризики можуть варіюватися від технічних збоїв і неочікуваних витрат до опору з боку персоналу та проблем з інтеграцією. Ігнорування або недооцінка цих потенційних загроз може призвести до значних затримок, перевитрат бюджету, зниження продуктивності та навіть повного провалу проєкту. Саме тому критично важливо заздалегідь ідентифікувати можливі перешкоди та розробити ефективні стратегії для їх мінімізації. Ключові ризики, що можуть виникнути під час впровадження розробленої систем, а також надамо практичні рекомендації щодо їх подолання представлено у таблиці 4.3.

Таблиця 4.3. Потенційні ризики при впровадженні системи та рекомендації щодо їх мінімізації

№	Потенційний ризик	Опис проблеми	Рекомендації щодо запобігання або мінімізації ризику
1	Перевантаження сервера при великій кількості запитів	Нестабільна робота або відмова системи при піковому навантаженні	Використання масштабованих хмарних рішень (Heroku, AWS, Render), кешування

2	Витік конфіденційної інформації	Відсутність захисту змінних оточення або логіки авторизації	Зберігання даних у `.env`, використання JWT, обмеження доступу до логів
3	Помилки при інтеграції платіжних систем	Невдалий або некоректний платіж, відсутність обробки помилок	Використання тестових режимів Stripe/LiqPay, обробка помилок API
4	Недостатнє тестування оновлень	Поява нових багів після розгортання	Використання staging-середовища, автоматизоване тестування, Git-гілки
5	Несумісність із мобільними пристроями	Некоректне відображення інтерфейсу на смартфонах	Повна адаптивність інтерфейсу, тестування в різних браузерах та роздільностях
6	Відсутність резервного копіювання	Втрата важливої інформації у разі збою	Налаштування регулярного резервного копіювання бази даних та завантажень
7	Складність у навчанні персоналу	Працівники не можуть ефективно працювати з системою	Проведення інструктажу, створення довідкової документації або відеоінструкцій
8	Правові ризики	Обробка персональних даних	Впровадження політики конфіденційності,

		без дотримання законодавчих вимог	погодження з умовами користування

Підсумовуючи, ефективне впровадження системи електронної комерції вимагає ретельного дотримання визначених вимог та системного підходу до мінімізації потенційних ризиків. Як було розглянуто в цьому розділі, успіх проєкту залежить не лише від технічної реалізації функціоналу, такого як каталог товарів, кошик, платіжні системи та адміністративна панель, а й від глибокого розуміння потреб різних категорій користувачів – адміністраторів, зареєстрованих клієнтів та гостьових відвідувачів.

Ключовим аспектом є чітке визначення функціональних і нефункціональних вимог, що гарантує відповідність розроблюваної системи бізнес-цілям та очікуванням користувачів. Паралельно з цим, проактивне виявлення та управління ризиками, від технічних збоїв до проблем з інтеграцією та опором змінам, є життєво важливим для запобігання затримкам та перевитратам бюджету. Впровадження системи моніторингу та аналітики (наприклад, LogRocket, Sentry, Google Analytics, Hotjar) забезпечує безперервне відстеження працездатності, виявлення помилок та збір даних для постійного вдосконалення функціоналу на основі реальної поведінки користувачів.

Таким чином, комплексний підхід до вимог та рекомендацій, що включає деталізацію функціоналу, врахування користувацьких сценаріїв, а також системне управління ризиками та моніторинг, закладає міцний фундамент для створення надійної, ефективної та успішної платформи електронної комерції, здатної відповідати сучасним викликам ринку та забезпечувати високий рівень задоволеності клієнтів.

На основі проведеної розробки та тестування веборієнтованої системи електронної комерції можна надати такі рекомендації щодо її впровадження та подальшої експлуатації:

По-перше, доцільно розміщувати систему на хостингу з підтримкою сучасних вебтехнологій, а саме – Node.js-середовища, PostgreSQL та

сертифікатів безпеки SSL/TLS. Це забезпечить стабільність роботи та захист персональних і платіжних даних користувачів.

По-друге, перед запуском у продакшн-середовище рекомендується провести стрес-тестування системи, особливо модулів обробки замовлень і оплати, щоб переконатися у їхній здатності витримувати навантаження в умовах пікового трафіку.

Також важливо приділяти увагу постійному оновленню пакунків, фреймворків і бібліотек, які використовуються у проєкті, щоб уникнути вразливостей безпеки та підтримувати актуальність технологій.

З метою зручності адміністрування й аналізу бізнес-метрик варто впровадити окремий розділ аналітики – для відстеження динаміки продажів, популярності товарів, поведінки користувачів тощо.

Крім того, для кращого користувацького досвіду рекомендовано реалізувати механізм кешування найпопулярніших запитів і сторінок, а також забезпечити мобільну оптимізацію або окремий застосунок для Android/iOS.

Розроблена система закладена у гнучку архітектуру, яка дозволяє легко масштабувати її функціональність відповідно до нових бізнес-потреб або технологічних вимог. Перспективними напрямками розвитку даної веборієнтованої системи можливі:

1. Інтеграція з мобільними платформами – створення окремих нативних або гібридних мобільних застосунків, які синхронізуються з основною базою даних та API.
2. Підтримка багатомовного інтерфейсу – впровадження мультимовності (наприклад, української, англійської та інших мов) для розширення цільової аудиторії.
3. Система рекомендацій на базі машинного навчання – розробка модуля персоналізованих товарних рекомендацій залежно від поведінки користувача, історії покупок та популярності товарів.
4. Підключення платіжних систем різних країн – інтеграція з локальними платіжними сервісами, що дасть змогу виходити на нові ринки.

5. Динамічне ціноутворення та акційні механізми – можливість задавати акції, знижки, купони, таймери розпродажів тощо.
6. Система повідомлень у реальному часі – реалізація інформування користувача про статус замовлення, зміну цін або залишків через WebSocket або push-нотифікації.
7. Інтеграція з службами доставки – автоматичне визначення вартості й часу доставки через API перевізників (наприклад, Нова Пошта, Укрпошта, Meest).

Реалізація вказаних напрямків дозволить перетворити систему на повноцінну комерційну платформу, конкурентоспроможну на сучасному ринку електронної торгівлі.

ВИСНОВКИ

У межах виконання дипломної роботи було розроблено повнофункціональну веборієнтовану систему електронної комерції, яка забезпечує повний цикл обслуговування користувача – від реєстрації до оформлення й оплати замовлення. У процесі роботи було здійснено всебічний аналіз предметної області, сформульовано функціональні та нефункціональні вимоги до системи, а також спроєктовано її архітектуру та логічну модель даних.

Розробка інформаційної бази базувалась на побудованій ER-моделі, яка охоплює всі ключові сутності системи: користувачів, товари, замовлення, кошик, платежі тощо. Для реалізації бази даних було обрано СУБД PostgreSQL, що забезпечує високу продуктивність, надійність і гнучкість у роботі з реляційними структурами.

Система була реалізована з використанням сучасного технологічного стеку: на клієнтському рівні застосовано бібліотеку React.js, на серверному – Node.js разом з Express.js, а для взаємодії з базою даних – ORM Sequelize. Таке поєднання дозволило створити масштабовану, модульну та безпечну систему. Програмні модулі були реалізовані згідно з принципами розділення відповідальностей і повторного використання коду, що підвищує зручність подальшої підтримки й розширення функціоналу.

Крім основного функціоналу, у систему було інтегровано зовнішні сервіси: платіжну систему для обробки онлайн-оплати, а також модуль електронних повідомлень, що дозволяє інформувати користувача про зміну статусу замовлення. Усі елементи були протестовані в інтегрованому середовищі, що підтвердило працездатність системи, її стабільність та відповідність початково сформульованим вимогам.

У результаті роботи було досягнуто поставлену мету – створити сучасну вебплатформу для електронної комерції з адаптивним дизайном,

структурованим інтерфейсом адміністратора, надійним збереженням даних і підтримкою основних бізнес-процесів. Отримані результати можуть бути використані як основа для реального впровадження в малий або середній бізнес, або як базова архітектура для подальшого розширення функціоналу, зокрема додавання підтримки мобільного застосунку, аналітики продажів чи багатомовного інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sivarajah, U., Irani, Z., Gupta, S., & Mahroof, K. (2020). Role of big data and social media analytics for business to business sustainability: A participatory web context. *Industrial Marketing Management*. https://doi.org/10.1016/J.INDMARMAN.2019.04.005.
2. Bresciani, S., Huarng, K., Malhotra, A., & Ferraris, A. (2021). Digital transformation as a springboard for product, process and business model innovation. *Journal of Business Research*. https://doi.org/10.1016/J.JBUSRES.2021.02.003.
3. Saarikko, T., Westergren, U., & Blomquist, T. (2020). Digital transformation: Five recommendations for the digitally conscious firm. *Business Horizons*. https://doi.org/10.1016/J.BUSHOR.2020.07.005.
4. Mann, G., Karanasios, S., & Breidbach, C. (2022). Orchestrating the digital transformation of a business ecosystem. *J. Strateg. Inf. Syst.*, 31, 101733. https://doi.org/10.1016/j.jsis.2022.101733.
5. Ardolino, M., Rapaccini, M., Saccani, N., Gaiardelli, P., Crespi, G., & Ruggeri, C. (2018). The role of digital technologies for the service transformation of industrial companies. *International Journal of Production Research*, 56, 2116 - 2132. https://doi.org/10.1080/00207543.2017.1324224.
6. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
7. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html

8. Pulasthi, L., Gunawardhana, D., & Gunawardhana, L. (2019). Process of Requirement Analysis Link to Software Development. *Journal of Software Engineering and Applications*. https://doi.org/10.4236/jsea.2019.1210025.
9. Kodubets, A., & Artemieva, I. (2021). Requirements Engineering for Software Systems: A Systematic Literature Review. *Programmnaya Ingeneria*. https://doi.org/10.17587/prin.12.339-349.
10. Yaseen, M. (2023). Empirical exploration of critical challenges of requirements implementation in global software development. *Journal of Software: Evolution and Process*, 36. https://doi.org/10.1002/smr.2604.
11. Umar, M., Lano, K., & Abubakar, A. (2025). Automated requirements engineering framework for agile model-driven development. *Frontiers in Computer Science*. https://doi.org/10.3389/fcomp.2025.1537100.
12. Melegati, J., Goldman, A., Kon, F., & Wang, X. (2019). A model of requirements engineering in software startups. *Inf. Softw. Technol.*, 109, 92-107. https://doi.org/10.1016/J.INFSOF.2019.02.001.
13. Kodubets, A., & Artemieva, I. (2021). Requirements Engineering for Software Systems: A Systematic Literature Review. *Programmnaya Ingeneria*. https://doi.org/10.17587/prin.12.339-349.

ДОДАТОК А

Фрагменти програмного коду. Контролера авторизації користувача

```
javascript
КопіюватиРедагувати
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { User } = require('../models'); // Sequelize модель користувача

// Секретний ключ для JWT (зберігається у .env файлі)
const JWT_SECRET = process.env.JWT_SECRET || 'super_secret_key';

// Реєстрація нового користувача
exports.register = async (req, res) => {
  try {
      const { email, password, name } = req.body;

      // Перевірка наявності користувача
      const existingUser = await User.findOne({ where: { email } });
      if (existingUser) {
      return res.status(400).json({ message: 'Користувач з таким email вже
існує' });
      }

      // Хешування пароля
      const hashedPassword = await bcrypt.hash(password, 10);

      // Створення користувача
      const user = await User.create({
      email,
      password_hash: hashedPassword,
      name,
      role: 'buyer',
      registration_date: new Date(),
      });

      res.status(201).json({ message: 'Користувач успішно зареєстрований' });
  } catch (error) {
      res.status(500).json({ message: 'Помилка сервера', error });
  }
};

// Вхід користувача
exports.login = async (req, res) => {
  try {
      const { email, password } = req.body;

      // Пошук користувача
      const user = await User.findOne({ where: { email } });
      if (!user) {
      return res.status(401).json({ message: 'Невірний email або пароль' });
      }

      // Перевірка пароля
      const isMatch = await bcrypt.compare(password, user.password_hash);
      if (!isMatch) {
      return res.status(401).json({ message: 'Невірний email або пароль' });
      }

      // Генерація JWT-токена
```

```
      const token = jwt.sign(
      { userId: user.user_id, role: user.role },
      JWT_SECRET,
      { expiresIn: '24h' }
      );

      res.status(200).json({
      message: 'Успішний вхід',
      token,
      user: {
      id: user.user_id,
      name: user.name,
      email: user.email,
      role: user.role,
      },
      });
  } catch (error) {
      res.status(500).json({ message: 'Помилка сервера', error });
  }
};
```