

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології») на тему:

«Вебплатформа управління магазином подарунків»

Виконала студентка групи ВТ-22-1

ХІМІЧ Ольга Олександрівна

Керівник роботи:

ГРОМСЬКИЙ Олександр Олександрович

Рецензент:

ЧИЖМОТРЯ Олексій Володимирович

Житомир - 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ

На кваліфікаційну роботу

Здобувач вищої освіти: **ХІМІЧ Ольга Олександрівна**

Керівник роботи: **ГРОМСЬКИЙ Олександр Олександрович**

Тема роботи: **«Вебплатформа управління подарунковим магазином»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані до роботи: розроблення вебплатформи управління магазином подарунків з метою автоматизації роботи з каталогом товарів, замовленнями та користувачами.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Громський О. О.	08.11.2025	10.12.2025
2	Громський О. О.	15.03.2026	16.03.2026
3	Громський О. О.	17.04.2026	19.04.2026

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел із 33 найменувань та одного додатка. Робота містить 35 рисунків, 9 таблиць та 7 лістингів. Повний обсяг роботи становить 93 сторінки, з них 84 сторінки основного тексту.

У роботі розроблено платформу для управління інтернет-магазином подарунків. Вона автоматизує основні бізнес-процеси електронної комерції та прискорює обробку замовлень. Систему побудовано за клієнт-серверною архітектурою з розділенням клієнтської та серверної частин, що дає змогу масштабувати її під час пікових навантажень, наприклад у святкові періоди, і зберігати швидкодію та безпеку транзакцій.

Серверну частину (RESTful API) написано мовою C# на платформі .NET Core. Клієнтський інтерфейс реалізовано як односторінковий застосунок (SPA) на фреймворку Vue.js зі стейт-менеджером Pinia. Каталог, замовлення та дані користувачів зберігаються в реляційній СУБД MS SQL Server. Бізнес-логіку перевірено багаторівневим тестуванням за допомогою Vitest, Playwright та Apache JMeter.

Додаток має адаптивний інтерфейс. Користувач може зареєструватися, переглядати товари, фільтрувати їх за приводом і бюджетом, користуватися кошиком та списком обраного (Wishlist) і оформлювати замовлення. Для адміністраторів передбачено окрему панель керування асортиментом, складськими залишками та статусами замовлень. Доступ до конфіденційних функцій захищено JWT-автентифікацією.

Ключові слова: ВЕБПЛАТФОРМА, ЕЛЕКТРОННА КОМЕРЦІЯ, ІНТЕРНЕТ-МАГАЗИН, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ТЕСТУВАННЯ.

					ІПЗ.КР.Б.-121-26-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розробник		Хіміч О.О.			Вебплатформа управління магазином подарунків	Літ.	Арк.
Керівник		Громський О.О.					Аркушів
							3
Рецензент		Чижмотря О.В.				Житомирська політехніка, група ВТ-22-1	
Зав. каф.		Вакалюк Т. А.					
							93

ABSTRACT

The qualification thesis consists of an introduction, three chapters, conclusions, a list of 33 references, and one appendix. It contains 35 figures, 9 tables, and 7 code listings. The total volume of the work is 93 pages, of which 84 pages are the main text.

The work presents a platform for managing an online gift shop. It automates the main e-commerce business processes and speeds up order processing. The system uses a client-server architecture that separates the client and server sides, which lets it scale during peak loads, such as holiday periods, while keeping response times fast and transactions secure.

The server side (a RESTful API) is written in C# on the .NET Core platform. The client interface is built as a single-page application (SPA) with the Vue.js framework and the Pinia state manager. The catalog, orders, and user data are stored in the MS SQL Server relational database. The business logic was tested at several levels using Vitest, Playwright, and Apache JMeter.

The application has a responsive interface. A user can register, browse products, filter them by occasion and budget, use the cart and wishlist, and place orders. Administrators have a separate panel for managing the product range, stock levels, and order statuses. Access to sensitive functions is protected by JWT authentication.

Keywords: CLIENT-SERVER ARCHITECTURE, E-COMMERCE, ONLINE STORE, TESTING, WEB PLATFORM.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ТА РЕАЛІЗАЦІЇ ПОДАРУНКОВОЇ ПРОДУКЦІЇ	10
1.1. Постановка задачі	10
1.2. Аналіз існуючих аналогів програмного продукту.....	14
1.3. Визначення архітектури та обґрунтування вибору інструментальних засобів.....	18
Висновки до першого розділу.....	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ УПРАВЛІННЯ МАГАЗИНОМ ПОДАРУНКІВ.....	27
2.1. Аналіз варіантів використання, функціональних вимог та об'єктно-орієнтованої структури.....	27
2.2. Розробка бази даних системи.....	42
2.3. Проєктування алгоритмів, архітектури RESTful API та бізнес-процесів	48
Висновки до другого розділу	55
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ	57
3.1. Інтерфейс користувача	57
3.2. Інтерфейс адміністратора та оформлення замовлення	61
3.3. Тестування платформи	66
Висновки до третього розділу	77
ВИСНОВКИ.....	80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	82
ДОДАТКИ.....	85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

ІПЗ – інженерія програмного забезпечення.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

ТЗ – технічне завдання.

ACID (Atomicity, Consistency, Isolation, Durability) – вимоги до надійних транзакцій бази даних: атомарність, узгодженість, ізолюваність, довговічність.

API (Application Programming Interface) – прикладний програмний інтерфейс.

CMS (Content Management System) – система керування вмістом.

CRUD (Create, Read, Update, Delete) – базові операції роботи з даними (створення, читання, оновлення, видалення).

CSRF (Cross-Site Request Forgery) – міжсайтова підробка запитів.

CSS (Cascading Style Sheets) – каскадні таблиці стилів.

CSV (Comma-Separated Values) – текстовий формат даних, розділених комами.

DOM (Document Object Model) – об’єктна модель документа.

E2E (End-to-End) – наскрізне тестування.

ERD (Entity-Relationship Diagram) – діаграма «сутність-зв’язок».

HTML (HyperText Markup Language) – мова гіпертекстової розмітки.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту.

JS (JavaScript) – об’єктно-орієнтована мова програмування.

JSON (JavaScript Object Notation) – текстовий формат обміну даними.

JWT (JSON Web Token) – відкритий стандарт для безпечної передачі інформації у вигляді JSON-об’єкта.

KPI (Key Performance Indicators) – ключові показники ефективності.

PDF (Portable Document Format) – переносний формат документів.

RBAC (Role-Based Access Control) – розмежування прав доступу на основі ролей користувачів.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						6

REST (Representational State Transfer) – архітектурний стиль для розподілених гіпермедійних систем.

RPS (Requests Per Second) – кількість запитів за секунду.

SEO (Search Engine Optimization) – оптимізація для пошукових систем.

SKU (Stock Keeping Unit) – одиниця складського обліку, унікальний артикул товару.

SPA (Single Page Application) – односторінковий вебзастосунок.

SQL (Structured Query Language) – мова структурованих запитів.

TCP (Transmission Control Protocol) – протокол керування передаванням даних.

UI (User Interface) – інтерфейс користувача.

UML (Unified Modeling Language) – уніфікована мова моделювання.

URL (Uniform Resource Locator) – уніфікований локатор (адреса) ресурсу.

UX (User Experience) – користувацький досвід.

XSS (Cross-Site Scripting) – міжсайтовий скриптинг.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						7

ВСТУП

Актуальність теми. Сучасний світ неможливо уявити без електронної комерції (e-commerce). Торгівля – один із найдавніших видів людської діяльності, який з’явився як результат розвитку продуктивних сил і необхідності обміну благами. Історично процеси купівлі-продажу вимагали фізичної присутності на ринках чи у крамницях та безпосереднього контакту між продавцем і покупцем.

Із стрімким розвитком цивілізації та інформаційних технологій змінювалися і самі формати торгівлі. З недавніх пір на заміну звичним фізичним магазинам приходять цифрові маркетплейси та онлайн-платформи. З кожним роком інтернет-шопінг отримує все більш широке поширення [4]. Цьому багато в чому сприяла глобальна діджиталізація та повсюдне використання мобільних пристроїв. Зараз багато людей працюють у віддаленому режимі і цінують свій час, тому популярністю користуються інтернет-покупки з адресною доставкою. Особливе місце на ринку займає ніша подарункової та сувенірної продукції, яка вимагає від платформ специфічного підходу: зручної категоризації за приводами, гнучкої фільтрації та можливості зберігати товари у «списах обраного».

Розробка та використання інтегрованого інтернет-магазину дають низку переваг. Клієнти отримують цілодобовий (24/7) доступ до каталогу товарів незалежно від місця перебування та можуть швидко шукати потрібне за категоріями, бюджетом і тематикою. Облік складських залишків і обробка замовлень переходять в автоматизовану цифрову форму, а операційні витрати бізнесу залишаються значно нижчими, ніж на утримання фізичних торгових точок. Оформлення покупок відбувається швидко, зручно й надійно, без прямих посередників, а особисті кабінети та списки вподобань (Wishlist) дають змогу персоналізувати користувацький досвід.

Метою кваліфікаційної роботи бакалавра є розробка інтегрованої вебплатформи інтернет-магазину подарункової продукції.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	8
		Громський О.О.				

Досягнення цієї мети передбачає кілька завдань. Передусім потрібно дослідити особливості роботи сучасних систем електронної комерції та вимоги до них, а також проаналізувати взаємодію користувачів (покупців та адміністраторів) із вебплатформою. Далі необхідно визначити клієнт-серверну архітектуру та структуру реляційної бази даних, обґрунтувати вибір сучасних засобів розробки (C# .NET Core, Vue.js, MS SQL Server) та спроектувати й розробити клієнтську і адміністративну частини платформи. Останнім завданням є комплексне автоматизоване тестування системи: модульне, інтеграційне, наскрізне та навантажувальне.

Об'єктом дослідження є процес створення та управління інформаційними системами у сфері електронної комерції.

Предметом дослідження є технології та методи розробки інтегрованої вебплатформи інтернет-магазину подарунків.

Методи дослідження. Для виконання роботи застосовано методи системного аналізу предметної області, об'єктно-орієнтованого аналізу та проєктування, моделювання за допомогою уніфікованої мови UML, методи проєктування реляційних баз даних, а також методи модульного, інтеграційного, наскрізного й навантажувального тестування програмного забезпечення. Інформаційною базою дослідження стали наукова та технічна література з інженерії програмного забезпечення й електронної комерції, офіційна документація використаних технологій і галузеві стандарти.

Практичне значення одержаних результатів полягає в тому, що розроблену вебплатформу доведено до рівня готового програмного продукту, придатного до впровадження в діяльність підприємств роздрібної торгівлі подарунковою та сувенірною продукцією. Створена система автоматизує ведення каталогу товарів, обробку замовлень і облік складських залишків, що знижує операційні витрати та підвищує якість обслуговування клієнтів.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						9

РОЗДІЛ 1. АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ТА РЕАЛІЗАЦІЇ ПОДАРУНКОВОЇ ПРОДУКЦІЇ

1.1. Постановка задачі

Особливості продажу подарункової продукції висувають додаткові вимоги до організації інтернет-магазину. Під час вибору подарунка користувачі часто орієнтуються не лише на назву або категорію товару, а й на привід, ціну, призначення, одержувача, доступність продукції та оцінки інших покупців. У зв'язку із цим каталог подарунків повинен мати зрозумілу структуру та забезпечувати можливість пошуку, фільтрації й сортування товарів за визначеними критеріями. Недостатньо продумана навігація або складний процес оформлення замовлення можуть призвести до відмови користувача від покупки та негативно вплинути на конкурентоспроможність магазину.

В умовах зростання кількості товарів і замовлень традиційні способи ведення обліку, що ґрунтуються на використанні паперових документів, окремих електронних таблиць, телефонних дзвінків і повідомлень у месенджерах, не забезпечують необхідної оперативності та достовірності інформації. Дані про товари, клієнтів і замовлення можуть зберігатися у різних джерелах, дублюватися або втрачати актуальність. Це ускладнює контроль залишків продукції, збільшує ймовірність помилок під час оформлення замовлень та потребує додаткових витрат робочого часу працівників.

Використання фрагментованих засобів обліку також не дає змоги сформувати цілісне уявлення про діяльність магазину. Наприклад, за відсутності централізованої інформаційної системи адміністратору складно визначити, які товари користуються найбільшим попитом, які позиції потребують поповнення, скільки замовлень перебуває на певному етапі опрацювання та наскільки ефективною є діяльність магазину загалом. Крім того, ручне перенесення інформації між різними файлами або програмами збільшує ризик виникнення суперечностей у даних.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						10

Актуальним є також питання забезпечення належного рівня користувацького досвіду. Інтерфейс вебплатформи повинен бути зрозумілим для різних категорій користувачів, забезпечувати логічну послідовність виконання операцій і коректно відображатися на пристроях із різними розмірами екранів. Особливе значення мають швидкість завантаження сторінок, доступність основних функцій, мінімальна кількість дій для оформлення замовлення та своєчасне інформування користувача про результати виконаних операцій.

Одним із ключових завдань є організація ефективного управління каталогом товарів. Кожна товарна позиція повинна містити назву, опис, ціну, зображення, інформацію про категорію, доступну кількість та інші характеристики, необхідні для прийняття рішення про покупку. Адміністратор повинен мати можливість створювати нові товари, редагувати наявні дані, змінювати ціну, контролювати кількість продукції та вилучати неактуальні позиції. Централізоване управління каталогом дозволяє підтримувати достовірність інформації та своєчасно оновлювати асортимент магазину.

Не менш важливим є управління складськими запасами. Інформація про кількість товару повинна автоматично враховуватися під час оформлення замовлення. Система не повинна дозволяти користувачеві замовити більшу кількість одиниць товару, ніж фактично доступно. Після підтвердження замовлення залишок відповідної продукції має оновлюватися. Такий підхід знижує ризик продажу відсутніх товарів і підвищує достовірність відомостей, представлених у каталозі.

Для забезпечення персоналізованої взаємодії з покупцями вебплатформа повинна підтримувати функції реєстрації, автентифікації та авторизації користувачів. Зареєстрований покупець має отримувати доступ до особистого кабінету, у якому зберігаються його контактні дані та інформація про сформовані замовлення. При цьому необхідно передбачити розмежування прав доступу між звичайними користувачами й адміністраторами. Клієнт повинен мати доступ лише до власних даних, тоді як адміністратор – до засобів керування товарами, категоріями, користувачами та замовленнями.

		Хімич О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						11

Особливу увагу необхідно приділити захисту персональних даних. Паролі користувачів не повинні зберігатися у відкритому вигляді, а доступ до захищених функцій системи має надаватися лише після успішного проходження автентифікації. Серверна частина вебплатформи повинна перевіряти вхідні дані, контролювати права доступу та коректно обробляти помилкові або потенційно небезпечні запити. Реалізація відповідних механізмів сприятиме зниженню ризику несанкціонованого доступу та втрати конфіденційної інформації.

Важливою складовою інтернет-магазину є процес формування та оформлення замовлення. Користувач повинен мати можливість додавати товари до кошика, змінювати їх кількість, вилучати окремі позиції та переглядати загальну вартість покупки. Перед завершенням оформлення система має перевіряти коректність зазначених даних і наявність обраної продукції. Після створення замовлення інформація про нього повинна зберігатися в базі даних і бути доступною користувачеві та адміністратору відповідно до їхніх повноважень.

Для підтримання довіри покупців до магазину доцільно передбачити систему оцінювання та відгуків. Відгуки дозволяють користувачам ділитися досвідом придбання продукції, а потенційним клієнтам – отримувати додаткову інформацію про її якість. На основі оцінок може формуватися середній рейтинг товару. Водночас механізм додавання відгуків повинен передбачати перевірку введених даних, ідентифікацію автора та можливість адміністрування опублікованого вмісту.

Управління замовленнями є одним із головних адміністративних процесів. Адміністратор повинен мати можливість переглядати сформовані замовлення, отримувати інформацію про їхній склад, контактні дані покупця, загальну вартість і поточний статус. Зміна статусу замовлення дає змогу відображати етапи його обробки, наприклад: «нове», «підтверджене», «опрацьовується», «відправлене», «виконане» або «скасоване». Централізований контроль таких станів спрощує організацію роботи магазину та зменшує ймовірність пропуску окремих замовлень.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						12

Окремою вимогою до програмної системи є забезпечення її стабільної роботи в умовах збільшення кількості одночасних запитів. Для магазинів подарункової продукції характерні сезонні коливання попиту, пов'язані зі святами та іншими подіями. У такі періоди навантаження на вебплатформу може суттєво зростати. Тому архітектура системи повинна забезпечувати раціональне використання ресурсів, ефективний обмін даними між клієнтською та серверною частинами, а також можливість подальшого масштабування.

Аналіз публікацій і практичного досвіду впровадження інформаційних систем у сфері роздрібної торгівлі свідчить, що цифровізація бізнес-процесів сприяє підвищенню оперативності обслуговування, зменшенню кількості помилок та покращенню контролю за рухом товарів і замовлень [6]. Інтегрована вебплатформа об'єднує інформацію про продукцію, клієнтів, товарні залишки, відгуки та операції продажу в єдиному інформаційному середовищі. Це створює умови для оптимізації діяльності працівників магазину й підвищення якості обслуговування покупців.

Таким чином, існує необхідність розроблення вебплатформи управління інтернет-магазином подарунків, яка поєднуватиме клієнтські та адміністративні функції, забезпечуватиме централізоване зберігання даних і автоматизуватиме основні процеси онлайн-торгівлі. Система має бути орієнтована як на покупців, які здійснюють пошук і замовлення подарункової продукції, так і на працівників магазину, відповідальних за підтримання асортименту, контроль залишків та опрацювання замовлень.

Кінцевим результатом має стати завершений вебзастосунок, який забезпечуватиме повний цикл взаємодії користувача з інтернет-магазином: від перегляду каталогу та вибору подарункової продукції до формування замовлення й контролю його стану. Для адміністратора система повинна надавати інструменти централізованого керування основними інформаційними об'єктами та бізнес-процесами магазину.

Отже, розроблення вебплатформи управління інтернет-магазином подарунків є актуальним інженерним завданням, вирішення якого передбачає

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	13
		Громський О.О.				

застосування сучасних методів проєктування програмного забезпечення, клієнт-серверної взаємодії, управління базами даних, захисту інформації та тестування. Реалізація запропонованої системи дозволить автоматизувати ключові операції онлайн-торгівлі, покращити якість обслуговування користувачів і створити технологічну основу для подальшого розвитку магазину.

1.2. Аналіз існуючих аналогів програмного продукту

Для формування вимог до майбутньої вебплатформи управління магазином подарунків було проведено аналіз трьох популярних рішень на українському ринку електронної комерції: Presentville, Gifty та UaMade. Кожна з цих платформ має свої архітектурні особливості, підходи до UI/UX дизайну та реалізації бізнес-логіки.

Інтернет-магазин Presentville (presentville.ua) [1]. Ця електронна торговельна платформа спеціалізується на роздрібній та гуртовій торгівлі оригінальними подарунками, сувенірною продукцією, аксесуарами, текстилем та різноманітними товарами з індивідуальними принтами й авторським дизайном. Загальний візуальний вигляд, архітектуру інтерфейсу та структуру навігації головної сторінки досліджуваного вебсайту представлено на рис. 1.1.

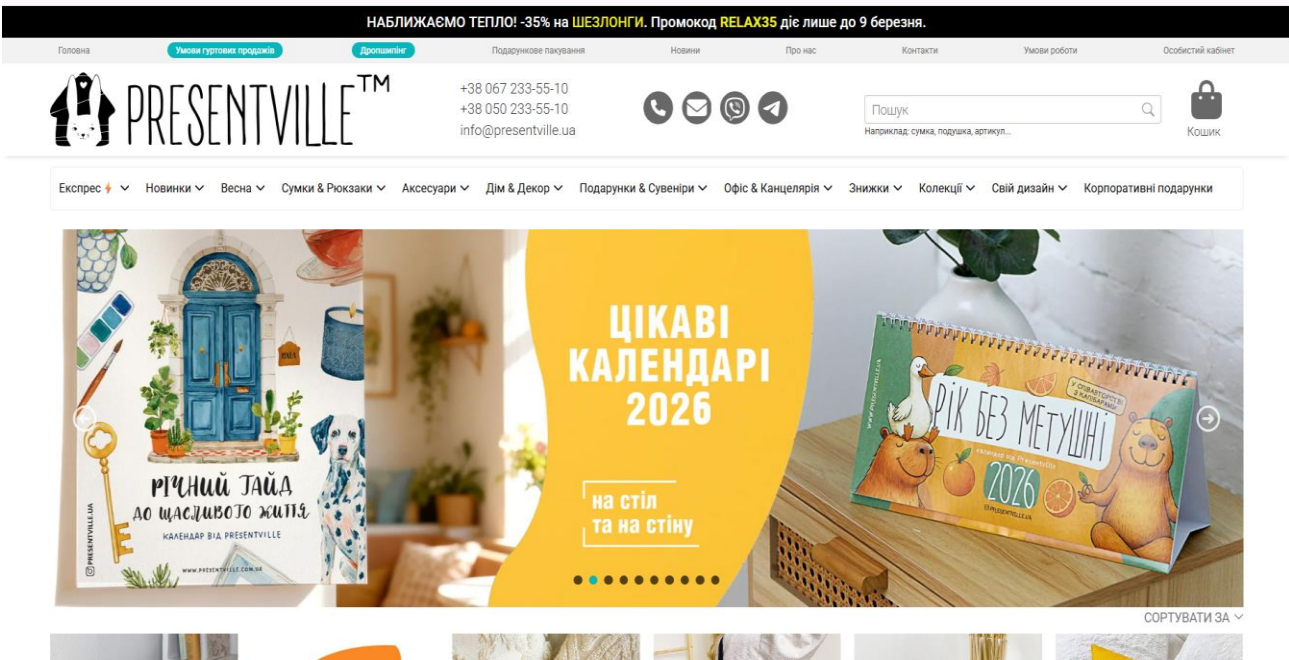


Рис. 1.1. Головне вікно Presentville

Переваги. Високий рівень UI/UX дизайну. Платформа використовує сучасний мінімалістичний підхід, де головний акцент робиться на високоякісному візуальному контенті. Процес оформлення замовлення (checkout) максимально спрощений і вимагає від користувача мінімальної кількості кліків.

Недоліки. Обмежена система фільтрації. Оскільки асортимент не є масовим, відсутні складні багатокритеріальні фільтри (наприклад, одночасний пошук за матеріалом, кольором та віком отримувача), що робить платформу менш масштабованою для великих обсягів різнотипних даних.

Бренд креативних подарунків Gifty (giftu.in.ua) [2].

Вузькоспеціалізований інтернет-магазин, який продає подарунки (планери, чашки, одяг) з ексклюзивними ілюстраціями від українських дизайнерів. Головне вікно вебзастосунку наведено на рис. 1.2.

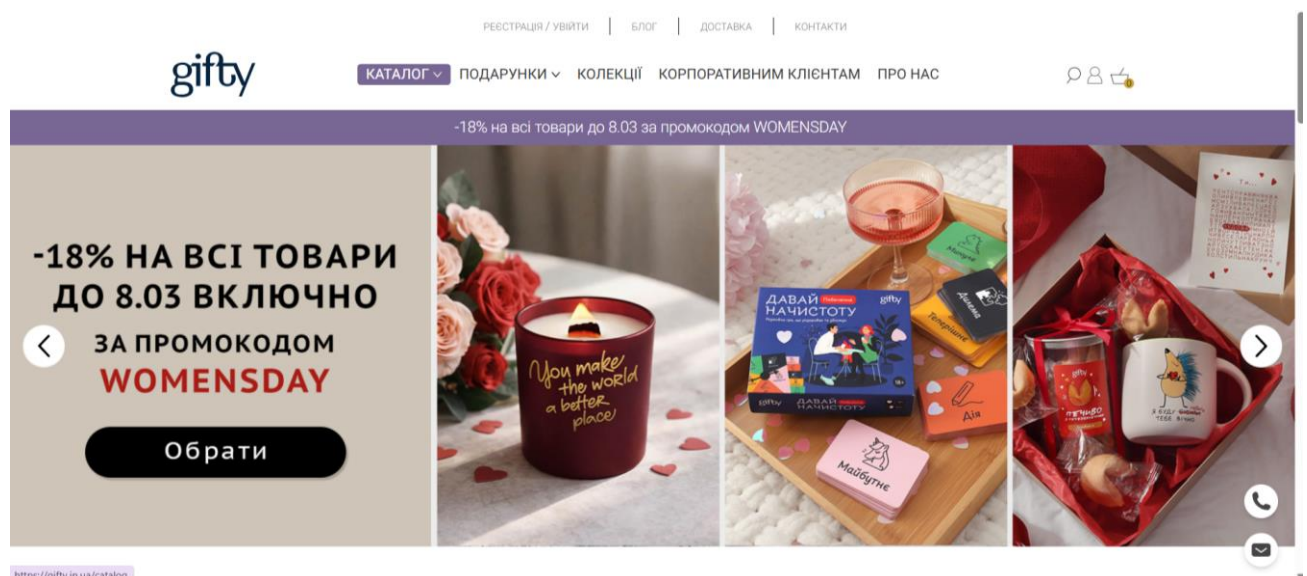


Рис. 1.2. Головне вікно Gifty.ua

Переваги. Дуже глибока та деталізована ієрархія каталогу, яка дозволяє шукати товари за конкретними святами, професіями чи родинними зв'язками. Важливою технічною перевагою є наявність вбудованого конструктора товарів, що дозволяє користувачу кастомізувати подарунок перед додаванням у кошик.

Недоліки. Дещо перевантажений інтерфейс користувача (UI). Велика кількість категорій та банерів на головній сторінці може розпорошувати увагу

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	15
		Громський О.О.				

користувача. Окрім цього, адаптивна (мобільна) версія сайту має простір для оптимізації швидкості завантаження елементів DOM.

Маркетплейс подарунків UaMade (uamade.ua). [3]

Даний ресурс функціонує як масштабна платформа-агрегатор, що інтегрує та об'єднує комерційні пропозиції від сотень автентичних українських крафтових виробників. З архітектурної та технічної точок зору така платформа є значно складнішою розподіленою системою, ніж класичний монолітний інтернет-магазин, оскільки потребує обробки великої кількості паралельних потоків даних та синхронізації залишків багатьох незалежних контрагентів. Загальний вигляд головної сторінки маркетплейсу представлено на рис. 1.3.

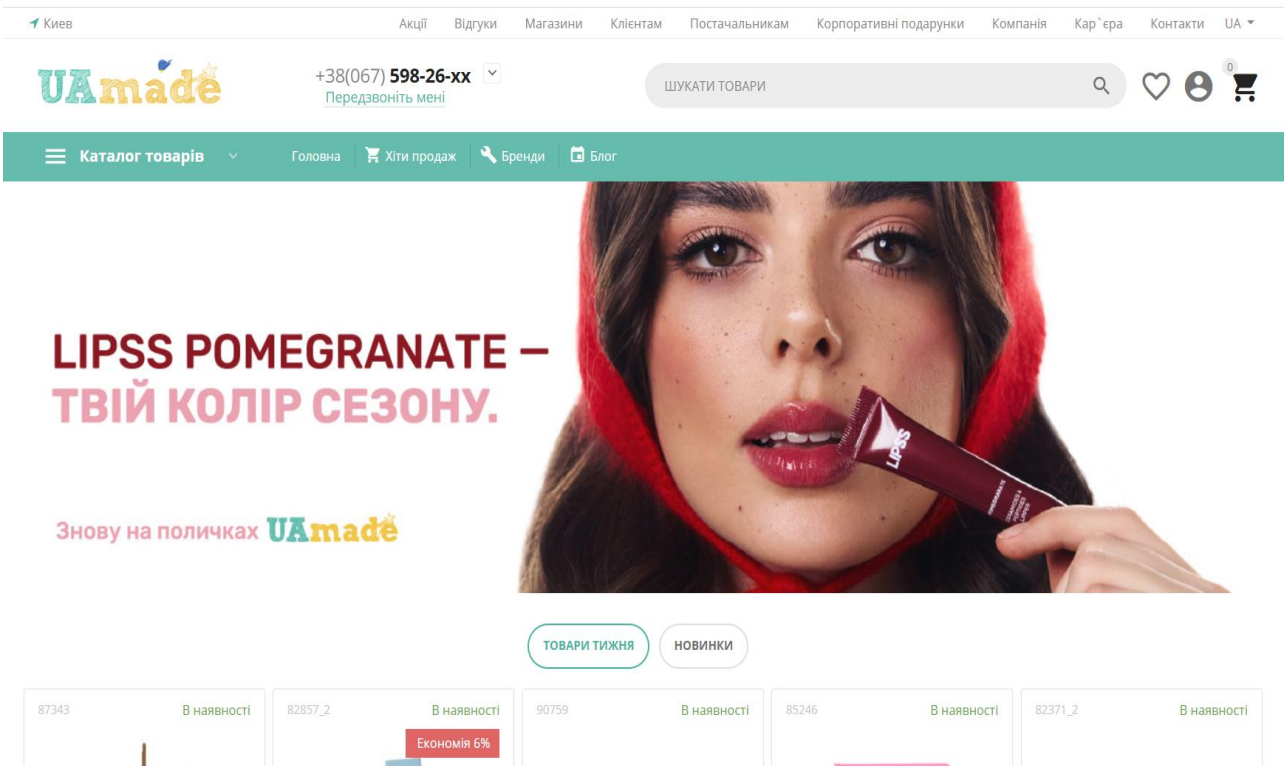


Рис. 1.3. Головне вікно UaMade

Переваги. Потужна система пошуку (на базі Elasticsearch або аналогів), яка швидко обробляє запити за великою базою даних. Розширена система фільтрації та тегування товарів. Реалізована складна архітектура зв'язків «Товар – Бренд (Постачальник) – Категорія».

Недоліки. Через те, що контент (фотографії, описи) надається різними постачальниками, сторінки каталогу виглядають візуально неоднорідно.

Складна архітектура збільшує навантаження на серверну частину (бекенд), що вимагає додаткових ресурсів для кешування сторінок.

Таблиця 1.1

Порівняння аналогів

Критерій порівняння	Presentville	Gifty	UaMade	Проектowana система
Спеціалізація	Сувеніри та подарунки з принтами	Дизайнерські та креативні речі	Маркетплейс крафтових товарів	Спеціалізований магазин подарунків
Система фільтрації	Деталізована (за святами, отримувачами)	Базова (лише за основними категоріями)	Розширена (теги, бренди, ціна, категорія)	Багатокритеріальна (за приводом, бюджетом, аудиторією)
Зручність інтерфейсу (UI/UX)	Перевантажені й елементами, складний	Мінімалістичний, інтуїтивно зрозумілий	Неоднорідний через різний контент продавців	Мінімалістичний, з акцентом на візуальний контент
Кастомізація подарунка	Високий рівень (вбудований конструктор)	Відсутня (готові дизайнерські рішення)	Обмежена (залежить від конкретного бренду)	Середній рівень (вибір пакування, додавання листівки)
Адаптивність (мобільна версія)	Підтримується, але має проблеми зі швидкістю	Високий рівень оптимізації	Підтримується, потребує оптимізації через обсяг даних	Повна адаптивність (Mobile-first підхід)
Особистий кабінет покупця	Присутній (історія замовлень)	Присутній (базовий функціонал)	Присутній (відстеження статусів, бонуси)	Розширений (відстеження статусу в реальному часі, збережені адреси)
Панель адміністратора	Закрита комерційна CMS	Закрита комерційна CMS	Складна система для багатьох постачальників	Кастомна (адаптована під конкретні бізнес-процеси магазину)

Розроблювана вебплатформа акумулює сильні сторони розглянутих аналогів: вона пропонуватиме таку ж потужну і гнучку систему фільтрації, як маркетплейси (UaMade), але при цьому збереже "чистий" та зрозумілий інтерфейс користувача, характерний для нішевих брендів (Gifty). Крім того, наявність кастомної панелі адміністратора дозволить оптимізувати бізнес-

процеси управління каталогом та замовленнями під конкретні потреби магазину подарунків.

1.3. Визначення архітектури та обґрунтування вибору інструментальних засобів

Для реалізації вебплатформи управління магазином подарунків доцільно застосувати клієнт-серверну архітектуру. Такий підхід широко використовується під час розроблення сучасних вебзастосунків електронної комерції, оскільки передбачає чітке розмежування функцій між компонентами системи, забезпечує централізоване опрацювання даних і створює умови для подальшого масштабування програмного продукту [11].

Клієнт-серверна архітектура передбачає, що користувач взаємодіє з клієнтською частиною вебплатформи, яка надсилає запити до серверної частини. Сервер опрацьовує отримані запити, виконує необхідні операції відповідно до бізнес-логіки, взаємодіє з базою даних і повертає клієнту результат. Завдяки такому підходу користувач не отримує прямого доступу до бази даних, а всі операції з інформацією виконуються через контрольовані програмні інтерфейси [23].

Архітектуру розроблюваної системи логічно поділено на три основні рівні: клієнтський рівень, серверний рівень і рівень даних. Загальну структуру програмного забезпечення наведено на рисунку 1.4.

Клієнтський рівень являє собою інтерфейс вебплатформи, з яким безпосередньо взаємодіє користувач. Залежно від визначеної ролі користувачем системи може бути покупець або адміністратор інтернет-магазину.

Для покупця клієнтська частина забезпечує перегляд каталогу подарунків, пошук і фільтрацію товарів, ознайомлення з описами, цінами, зображеннями та відгуками, додавання вибраних позицій до кошика, оформлення замовлення й роботу з особистим кабінетом. Для адміністратора клієнтський рівень надає інтерфейс керування товарами, категоріями, замовленнями, складськими залишками, користувачами та іншими об'єктами системи.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						18

До основних завдань клієнтського рівня належать відображення візуального вмісту вебплатформи (каталогу подарунків, зображень, описів товарів, цін, рейтингів і повідомлень про результати операцій), отримання введених користувачем даних (параметрів пошуку та фільтрації, реєстраційних і контактних відомостей, інформації для оформлення замовлення й текстів відгуків), а також формування та надсилання HTTP-запитів до серверної частини для отримання, створення, оновлення чи видалення інформації. Крім того, клієнтський рівень динамічно оновлює вміст сторінки без її повного перезавантаження (наприклад, після додавання товару до кошика чи застосування фільтра), виконує попередню перевірку коректності введених даних, зокрема заповнення обов'язкових полів, формату електронної пошти та допустимості числових значень, і адаптує інтерфейс до різних розмірів екрана для зручної роботи на персональних комп'ютерах, планшетах та мобільних пристроях.

Валідація на клієнтській стороні покращує користувацький досвід і дозволяє своєчасно повідомляти про помилки введення. Водночас вона не може бути єдиним механізмом перевірки даних, оскільки клієнтські перевірки можуть бути обійдені. Тому всі критично важливі дані повинні повторно перевірятися серверною частиною.

Серверний рівень є центральною частиною вебплатформи та відповідає за реалізацію її основної бізнес-логіки. Він приймає запити від клієнтського застосунку, перевіряє їхню коректність, виконує необхідні операції, взаємодіє з базою даних і формує відповідь у стандартизованому форматі.

Взаємодію між клієнтською та серверною частинами доцільно реалізувати за допомогою REST API. У такому разі доступ до ресурсів системи здійснюється через визначені кінцеві точки, а для виконання операцій застосовуються стандартні методи протоколу HTTP. Метод GET використовується для отримання даних, POST – для створення нових ресурсів, PUT або PATCH – для оновлення наявних даних, DELETE – для їх видалення [10].

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						19

До основних завдань серверного рівня належать приймання та маршрутизація HTTP-запитів від клієнтського застосунку, автентифікація користувачів і перевірка їхніх облікових даних під час входу, а також авторизація та контроль прав доступу до окремих функцій й інформаційних ресурсів. Сервер генерує, передає та перевіряє токени доступу або застосовує інший механізм підтримання автентифікованого сеансу, реалізує бізнес-правила інтернет-магазину (перевірку наявності товарів, обчислення вартості кошика, застосування знижок чи промокодів, створення замовлень і змінення їхніх статусів), перевіряє коректність отриманих від клієнта даних та обробляє виняткові ситуації. Окрім того, серверний рівень забезпечує роботу з файлами, зокрема завантаження, перевірку, збереження та отримання зображень товарів, формує структуровані відповіді у форматі JSON, взаємодіє з базою даних для створення, читання, оновлення й видалення інформації та реєструє важливі операції й помилки, що виникають під час роботи вебплатформи.

На серверному рівні також повинні реалізовуватися механізми захисту даних. Зокрема, необхідно передбачити хешування паролів, перевірку прав доступу, захист кінцевих точок API, обмеження некоректних запитів і безпечне опрацювання даних, отриманих від користувачів. Сервер не повинен довіряти інформації, яка надходить із клієнтської частини, без її додаткової перевірки.

Рівень даних відповідає за централізоване зберігання, пошук, додавання, оновлення та видалення інформації, необхідної для функціонування інтернет-магазину. Для розроблюваної вебплатформи доцільно використати реляційну базу даних, оскільки предметна область характеризується чітко структурованими сутностями та наявністю логічних зв'язків між ними.

Основними сутностями бази даних є користувачі, ролі, товари, категорії, замовлення, позиції замовлень, відгуки, складські залишки та промокоди. Наприклад, один користувач може сформувати декілька замовлень, кожне замовлення може містити декілька товарів, а один товар належить до визначеної категорії та може мати декілька відгуків.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						20

Використання реляційної моделі даних забезпечує чітке визначення структури інформаційних сутностей, підтримання зв'язків між таблицями за допомогою первинних і зовнішніх ключів та зменшення дублювання інформації завдяки нормалізації структури бази даних. Вона також гарантує цілісність і узгодженість даних, дає змогу виконувати вибірки із застосуванням фільтрації, сортування, групування та об'єднання таблиць і підтримує транзакції під час виконання взаємопов'язаних операцій [13, 14].

Особливе значення має використання транзакцій під час оформлення замовлення. Створення замовлення, додавання його товарних позицій і зміна складських залишків повинні розглядатися як єдина операція. Якщо на одному з етапів виникає помилка, усі попередньо виконані зміни мають бути скасовані. Це дозволяє запобігти появі неповних замовлень або некоректних значень кількості товарів.

Звернення до бази даних виконується лише серверною частиною. Клієнтський застосунок не має прямого доступу до таблиць та облікових даних системи керування базами даних. Такий підхід дозволяє централізовано контролювати операції, перевіряти права користувачів і зменшувати ризик несанкціонованого доступу.

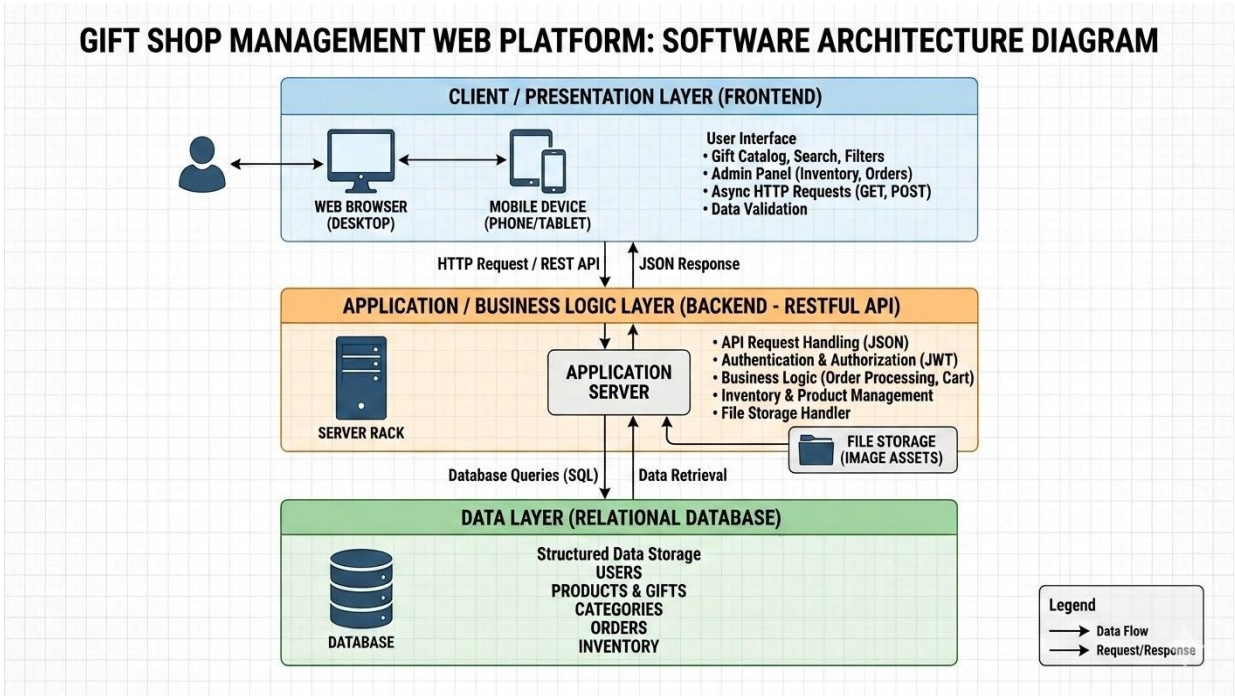


Рис. 1.4. Архітектура програмного забезпечення

Взаємодія між компонентами вебплатформи відбувається відповідно до циклу «запит – відповідь». Користувач виконує певну дію в інтерфейсі, після чого клієнтський застосунок формує HTTP-запит і надсилає його до відповідної кінцевої точки REST API. Серверна частина приймає запит, перевіряє передані параметри, виконує бізнес-логіку, звертається до бази даних і повертає результат клієнту. Після отримання відповіді клієнтський застосунок оновлює відповідну частину інтерфейсу.

Наприклад, під час застосування фільтра «Подарунки на день народження вартістю до 1000 грн» клієнтський застосунок формує GET-запит, що містить ідентифікатор або назву відповідної категорії, привід і максимальне значення ціни. Сервер приймає цей запит, перевіряє отримані параметри та формує звернення до рівня даних. Після виконання запиту база даних повертає перелік товарів, які відповідають установленим критеріям.

Серверна частина перетворює отримані результати на об'єкти передавання даних, формує JSON-відповідь і надсилає її клієнтському застосунку. Після отримання відповіді клієнтська частина динамічно оновлює каталог та відображає користувачеві відібрані товари без повного перезавантаження вебсторінки.

Іншим прикладом є оформлення замовлення. Клієнтська частина передає серверу відомості про вибрані товари, їхню кількість і контактні дані покупця. Сервер повторно перевіряє актуальні ціни та наявність кожного товару, обчислює загальну вартість, створює замовлення і відповідні позиції в базі даних, після чого оновлює складські залишки. У разі успішного виконання операції клієнт отримує підтвердження та ідентифікатор створеного замовлення.

Поділ програмного забезпечення на незалежні логічні рівні забезпечує модульність системи. Клієнтську та серверну частини можна розробляти й тестувати окремо за умови дотримання погодженого контракту API. Зміни в реалізації інтерфейсу не потребують безпосереднього втручання у структуру бази даних, а модифікація способу зберігання інформації не повинна впливати на клієнтський застосунок за умови незмінності формату серверних відповідей.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						22

Обрана архітектура також сприяє забезпеченню безпеки вебплатформи, оскільки клієнт не взаємодіє з базою даних безпосередньо. Серверна частина виступає контрольованим проміжним рівнем, який виконує перевірку вхідних даних, прав доступу та відповідності запитів установленим бізнес-правилам.

Отже, застосування трирівневої клієнт-серверної архітектури є обґрунтованим рішенням для розроблення вебплатформи управління магазином подарунків. Вона забезпечує розмежування відповідальності між компонентами, централізоване опрацювання даних, можливість незалежного розвитку окремих частин системи, підвищення рівня захищеності та створює передумови для масштабування програмного забезпечення відповідно до зростання кількості користувачів, товарів і замовлень.

Для успішної реалізації проєкту вебплатформи управління магазином подарунків необхідно обрати оптимальний стек технологій, який забезпечить швидку розробку, надійну роботу системи та зручність її подальшої підтримки. Вибір інструментальних засобів здійснювався з урахуванням трирівневої архітектури додатка.

Для реалізації серверної частини (RESTful API) та обробки бізнес-логіки обрано платформу ASP.NET Core [15] і мову програмування C# [16]. Цей вибір зумовлений високою продуктивністю фреймворку, який є одним із найшвидших сучасних вебфреймворків, що критично важливо для інтернет-магазинів із потенційно великою кількістю одночасних запитів. Не менш важливими є безпека та надійність: строга типізація C# зменшує кількість помилок ще на етапі розробки, а вбудовані механізми фреймворку захищають від поширених кібератак (CSRF, XSS) і забезпечують зручну систему авторизації (ASP.NET Core Identity). Окрім того, архітектура фреймворку дає змогу легко масштабувати функціонал магазину, додаючи нові модулі, наприклад інтеграцію з платіжними системами. Подальший вибір інструментальних засобів здійснювався на основі порівняльного аналізу сучасних рішень для кожного рівня трирівневої архітектури.

Для розробки користувацького інтерфейсу вітрини магазину та панелі адміністратора розглядалися дві найпопулярніші сучасні JavaScript-технології: React.js та Vue.js. React.js є потужною бібліотекою, розробленою компанією Facebook; її перевагами є велика екосистема, компонентний підхід і використання Virtual DOM для швидкого рендерингу інтерфейсу. Проте React.js потребує значної кількості сторонніх бібліотек для управління станом (наприклад, Redux) і маршрутизації, що підвищує поріг входження та ускладнює початкову конфігурацію проєкту. Vue.js також спирається на компонентний підхід і Virtual DOM, але пропонує інтегрованіше «коробкове» рішення для створення односторінкових застосунків (SPA), вирізняється інтуїтивно зрозумілим синтаксисом, високою швидкістю роботи та офіційною підтримкою інструментів екосистеми [28].

Враховуючи вимоги до платформи інтернет-магазину, де критично важливим є швидке та реактивне оновлення інтерфейсу, наприклад під час роботи з кошиком товарів, фінальний вибір було зроблено на користь Vue.js. У поєднанні зі стейт-менеджером Pinia ця технологія дає змогу ефективно керувати станом кошика, забезпечує високу реактивність інтерфейсу та чітко відокремлює логіку представлення від бізнес-логіки [17].

Дані інтернет-магазину, зокрема профілі клієнтів, історія замовлень, ієрархія категорій та атрибути подарунків, мають чітку структуру й багато взаємозв'язків, тому для рівня зберігання даних порівнювалися дві реляційні СУБД: MS SQL Server та PostgreSQL. MS SQL Server є потужною комерційною СУБД від Microsoft, яка гарантує цілісність даних (ACID), демонструє високу швидкість виконання складних SQL-запитів та ідеально інтегрується з екосистемою .NET, проте має вищу вартість ліцензування повноцінних версій і вимогливіша до апаратних ресурсів сервера. PostgreSQL є об'єктно-реляційною СУБД з відкритим вихідним кодом, яка суворо відповідає стандартам SQL, підтримує транзакційність, добре масштабується, безкоштовна для комерційного використання й зручно працює з форматом JSON. Для забезпечення цілісності даних та ефективного зв'язку між сутностями магазину (товарами, категоріями,

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						24

користувачами, замовленнями) було обрано MS SQL Server, що гарантує надійність, високу швидкість складних запитів і тісну інтеграцію з обраною платформою .NET [27].

Висновки до першого розділу

У першому розділі було проведено комплексний аналіз предметної області, досліджено специфіку ринку електронної комерції та сформовано ключові вимоги до проєктуваної вебплатформи управління магазином подарунків.

Детальне дослідження галузевих тенденцій показало стабільне зростання сегмента онлайн-торгівлі, що вимагає від бізнесу впровадження гнучких автоматизованих рішень. Традиційні методи управління вже не здатні впоратися з викликами категоризації, контролем складських запасів та піковими навантаженнями у святкові періоди. Установлено, що впровадження спеціалізованого вебзастосунку є економічно доцільним, оскільки дає змогу автоматизувати обробку замовлень, підвищити точність обліку складських залишків та поліпшити якість обслуговування клієнтів.

Для формування вимог до UI/UX дизайну та бізнес-логіки було здійснено порівняльний аналіз популярних українських платформ: Presentville, Gifty та UaMade. Аналіз виявив їхні сильні та слабкі сторони, що дозволило визначити стратегію для нашого проєкту: майбутня система повинна поєднувати потужну багатокритеріальну систему фільтрації (як у великих маркетплейсів) із чистим, мінімалістичним та інтуїтивно зрозумілим інтерфейсом.

На основі сформованих вимог було визначено та обґрунтовано архітектуру програмного забезпечення. Найбільш доцільним рішенням стало використання трирівневої клієнт-серверної архітектури. Такий розподіл на клієнтський рівень, серверний рівень та рівень даних гарантує чітке розділення відповідальності, високу безпеку та можливість незалежного масштабування кожного з компонентів.

Серверна частина (Бекенд): побудована на базі високопродуктивного фреймворку ASP.NET Core (C#), який забезпечує надійну обробку бізнес-логіки, сувору типізацію та вбудовані механізми захисту.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						25

Клієнтська частина (Фронтенд): реалізовуватиметься за допомогою прогресивного фреймворку Vue.js та стейт-менеджера Pinia. Це дозволить створити реактивне Single Page Application (SPA) з миттєвим оновленням інтерфейсу, що є критично важливим для роботи з кошиком та фільтрами.

Рівень зберігання даних: обрано комерційну реляційну СУБД MS SQL Server, яка гарантує цілісність даних (ACID), чудово справляється зі складними зв'язками між сутностями магазину та тісно інтегрується з платформою .NET.

Таким чином, у першому розділі було повністю обґрунтовано доцільність створення продукту, його архітектуру та набір інструментів. Отримані результати та прийняті рішення формують надійний фундамент для наступного етапу – детального проєктування бази даних, алгоритмів роботи платформи та розробки діаграм взаємодії.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ УПРАВЛІННЯ МАГАЗИНОМ ПОДАРУНКІВ

2.1. Аналіз варіантів використання, функціональних вимог та об'єктно-орієнтованої структури

На етапі проєктування логіки функціонування вебплатформи управління магазином подарунків критично важливим є визначення основних суб'єктів взаємодії (акторів) та сценаріїв їхньої поведінки в системі. На основі аналізу бізнес-вимог до розроблюваного додатку було виділено три ключові архітектурні ролі: Гість (неавторизований користувач), Покупець (авторизований клієнт) та Адміністратор [7].

Для формалізації вимог до функціональності системи та візуалізації зв'язків між акторами й прецедентами було побудовано діаграму варіантів використання (Use Case Diagram) за методологією уніфікованої мови моделювання UML [8]. Загальний вигляд розробленої діаграми прецедентів представлено на рис. 2.1.

Детальний аналіз побудованої UML-діаграми (див. рис. 2.1) дозволяє визначити ієрархію ролей та специфіку їхньої взаємодії з програмними модулями платформи:

Актор «Гість» відображає базовий рівень доступу. Його прецеденти охоплюють ознайомчий функціонал сайту: перегляд головної сторінки, навігацію каталогом, використання підсистеми багатокритеріальної фільтрації (сортування за бюджетом, приводом чи категорією), а також ініціалізацію сесії авторизації або реєстрації.

Актор «Покупець» знаходиться у відношенні узагальнення (generalization / успадкування) до актора «Гість». Це означає, що авторизований клієнт автоматично переймає всі базові можливості неавторизованого користувача, проте отримує розширені права доступу до ядра електронної комерції (управління кошиком, Checkout, доступ до логів особистого кабінету та редагування персонального профілю).

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	27
		Громський О.О.				

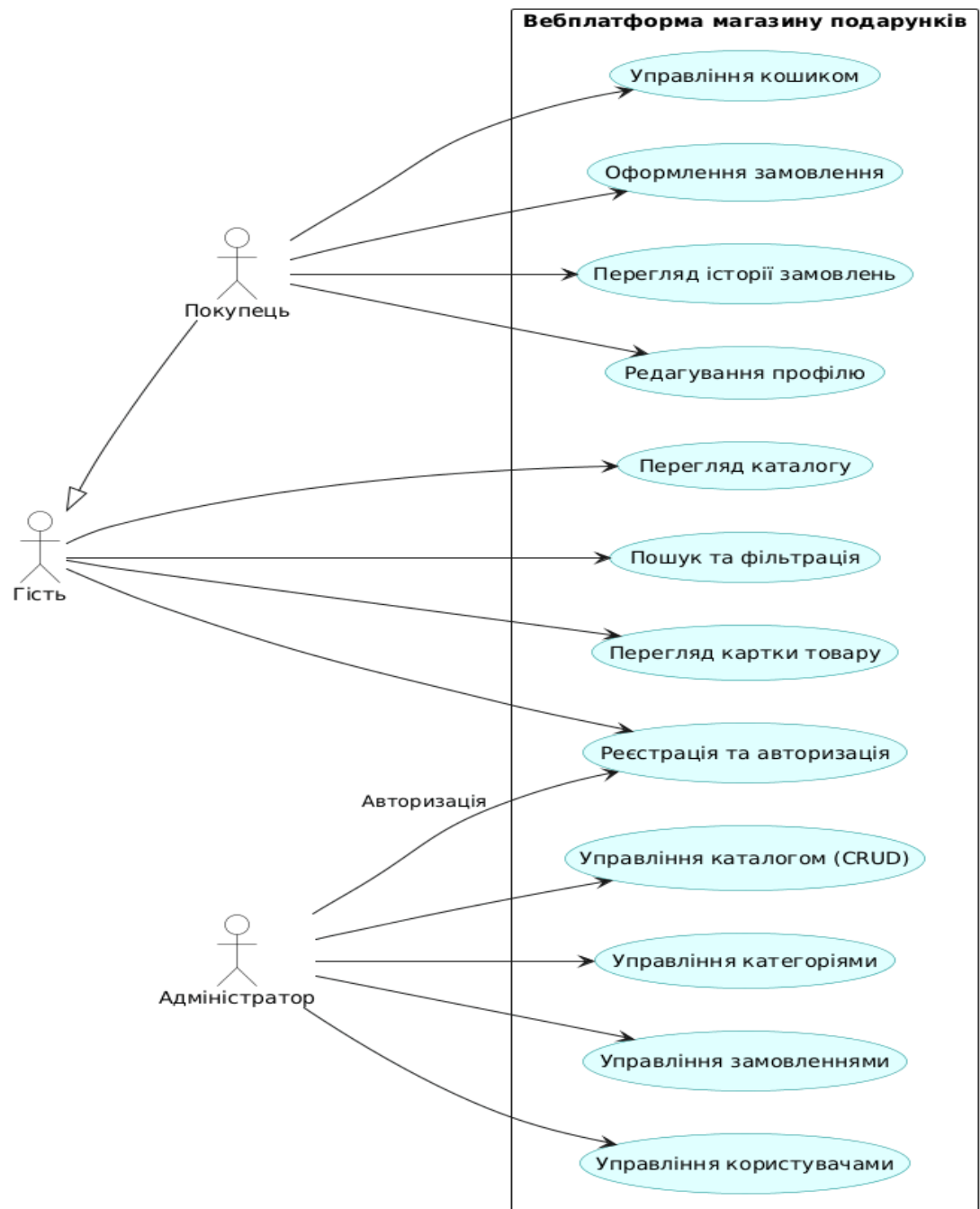


Рис. 2.1. Діаграма варіантів використання

Актор «Адміністратор» наділений найвищим рівнем привілеїв (модерація контенту, керування номенклатурою товарів, зміна статусів замовлень та аналітика продажів). Його інтерфейс є відокремленим від клієнтської логіки, що підвищує загальний рівень безпеки вебплатформи. Адміністративний доступ розмежовано за моделлю ролей (RBAC): окрім ролі «Адміністратор» із повними правами, передбачено ролі «Менеджер» (опрацювання замовлень і контенту),

«Модератор» (модерація відгуків) та «Працівник складу» (облік товарних залишків), тоді як звичайні авторизовані користувачі мають роль «Покупець».

Побудована Use Case діаграма слугує безпосереднім інженерним підґрунтям для подальшого проєктування структури бази даних (схеми ERD) та розробки архітектурних компонентів програмного забезпечення (класів та контролерів вебдодатка) [24].

Підсистема «Користувачі та безпека» (User Management Module) відповідає за процеси автентифікації, авторизації, управління профілями клієнтів та розмежування прав доступу між різними категоріями користувачів.

Гість (неавторизований користувач) повинен мати можливість створити новий обліковий запис, указавши ім'я, адресу електронної пошти, номер телефону та пароль, авторизуватися в системі за допомогою електронної пошти й пароля, а також ініціювати відновлення пароля через його скидання на електронну пошту у разі втрати.

Покупець (авторизований клієнт) повинен мати доступ до особистого кабінету для перегляду персональних даних, можливість редагувати контактну інформацію та додавати чи видаляти збережені адреси доставки для швидшого оформлення майбутніх замовлень, змінювати поточний пароль, безпечно завершувати сеанс роботи й переглядати історію залишених ним відгуків.

Адміністратор (керуючий магазином) повинен мати можливість авторизуватися в закритій панелі управління за спеціальними обліковими даними, переглядати загальний список зареєстрованих покупців, шукати користувачів за ім'ям, електронною поштою або номером телефону, переглядати детальну інформацію про конкретного клієнта (контакти, дату реєстрації, загальну суму покупок) та блокувати чи деактивувати облікові записи клієнтів, які порушують правила платформи.

На рівні автоматизованих процесів система повинна валідувати введені дані на стороні сервера, зокрема перевіряти унікальність електронної пошти при реєстрації та складність пароля, хешувати паролі користувачів (наприклад, алгоритмом BCrypt) перед збереженням у базі даних MS SQL, оскільки

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						29

зберігання паролів у відкритому вигляді заборонено, генерувати та повертати JWT при успішній авторизації для захисту подальших API-запитів, автоматично відхиляти запити до захищених маршрутів у разі недійсного або простроченого токена та надсилати email-сповіщення під час реєстрації нового користувача чи запиту на скидання пароля.

Підсистема «Каталог подарунків» забезпечує відображення вітрини інтернет-магазину, пошук і фільтрацію товарів та управління контентом з боку адміністрації.

Гість повинен мати можливість переглядати загальний каталог усіх доступних подарунків із розбиттям на сторінки або динамічним підвантаженням, шукати товари за назвою, ключовими словами чи артикулом (SKU) та застосовувати комплексні фільтри за категорією, приводом, отримувачем, ціновим діапазоном і наявністю на складі. Він також повинен мати змогу сортувати відфільтровані товари за ціною, популярністю або новизною та переглядати детальну картку обраного подарунка, яка містить назву, опис, ціну, поточний статус наявності та галерею зображень.

Покупець успадковує всі можливості гостя, а додатково може залишати текстові відгуки й виставляти числову оцінку (рейтинг від 1 до 5) товарам, які раніше придбав, та додавати подарунки до списку бажаного (Wishlist) для подальшого перегляду чи швидкої покупки.

Адміністратор повинен мати можливість додавати нові товари із зазначенням їхніх атрибутів (назви, артикулу, ціни, детального опису), редагувати наявні товари або приховувати їх з публічного каталогу, керувати медіаконтентом товару (завантажувати кілька зображень, видаляти їх і призначати головне), створювати, редагувати та видаляти ієрархічні категорії, керувати тегами-довідниками для специфічних фільтрів магазину (додавати нові приводи й отримувачів та прив'язувати їх до товарів), а також модерувати відгуки покупців перед публікацією.

На рівні автоматизованих процесів система повинна розраховувати й оновлювати середній рейтинг кожного товару на основі залишених оцінок,

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	30
		Громський О.О.				

автоматично оновлювати статус наявності товару в картці, оптимізувати завантажені зображення (стискати та конвертувати у сучасні формати, наприклад WebP) для пришвидшення рендерингу сторінок і формувати SEO-дружні URL-адреси (slugs) для карток товарів та категорій на основі їхніх назв [5].

Підсистема «Кошик та Замовлення» (Order Processing Module) відповідає за весь життєвий цикл покупки: від додавання першого подарунка до кошика до отримання товару клієнтом і закриття замовлення.

Гість повинен мати можливість додавати обрані товари до віртуального кошика, переглядати його вміст, змінювати кількість одиниць товару або видаляти позиції, причому система зберігає вміст кошика гостя (наприклад, у LocalStorage чи cookie) протягом визначеного часу, щоб дані не втрачалися при оновленні сторінки. Для переходу до оформлення замовлення гість повинен пройти швидку реєстрацію або авторизацію, перейшовши в роль покупця.

Покупець повинен мати можливість оформити замовлення, заповнивши форму з контактними даними та адресою доставки або обравши збережену адресу, обрати спосіб доставки (кур'єром, самовивозом чи через відділення «Нової Пошти») і спосіб оплати (онлайн-карткою або післяплатою), додати коментар до замовлення, застосувати промокод на знижку перед підтвердженням суми та скасувати замовлення, якщо воно ще перебуває у статусі «Нове» або «В обробці».

Адміністратор повинен отримувати миттєві сповіщення про надходження нових замовлень, переглядати повну інформацію про замовлення (перелік товарів, суму, дані клієнта, спосіб доставки та коментарі), вручну змінювати статуси замовлення в міру його виконання, редагувати склад замовлення або адресу доставки на прохання клієнта, поки замовлення не відправлено, та скасовувати замовлення із зазначенням причини.

На рівні автоматизованих процесів система повинна розраховувати підсумкову вартість кошика в реальному часі з урахуванням ціни товарів, їхньої кількості та застосованих знижок, генерувати унікальний номер для кожного

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	31
		Громський О.О.				

нового замовлення, ініціювати транзакцію перенесення товарів із сутності CartItem до OrderItem з фіксацією ціни на момент покупки, тимчасово резервувати товари на складі (reservedStock) на час онлайн-оплати, надсилати покупцеві email з підтвердженням замовлення та електронним чеком і інтегруватися з API платіжних шлюзів (наприклад, LiqPay чи Fondy) для безпечної обробки платежів та автоматичної зміни статусу на «Оплачено».

Підсистема «Управління складом та запасами» (Inventory & Warehouse Module) автоматизує облік товарів, взаємодію з постачальниками, контроль залишків та інвентаризацію, що є критично важливим для безперебійної роботи магазину подарунків.

Гість і покупець не мають прямого доступу до функцій управління складом; їхня взаємодія обмежується візуальним відображенням статусу товару на вітрині, наприклад «В наявності», «Закінчується: залишилось 2 шт.» або «Немає в наявності», який генерує ця підсистема.

Адміністратор повинен мати можливість переглядати поточний стан складу у вигляді зведеної таблиці з артикулами (SKU), назвами товарів та їхніми фізичними й доступними залишками, вести базу постачальників (додавати компанії, редагувати контактні дані, видаляти неактуальних партнерів), формувати електронні документи «Замовлення постачальнику» (Purchase Order), проводити оприбуткування товару, що автоматично збільшує залишки, здійснювати ручне списання товарів із обов'язковим зазначенням причини та переглядати детальну історію руху кожної одиниці товару.

На рівні автоматизованих процесів система повинна вираховувати доступний для продажу залишок за формулою $\text{Available Stock} = \text{Current Stock} - \text{Reserved Stock}$, блокувати додавання товару в кошик за нульового залишку, безперервно логувати будь-які зміни кількості (створюючи записи StockTransaction при продажу, оприбуткуванні чи списанні), відстежувати критичні залишки (reorderLevel) і сповіщати адміністратора, коли кількість популярного товару падає нижче заданого мінімуму, та автоматично скасовувати

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						32

резервування й повертати товар у вільний продаж, якщо покупець не завершив оплату протягом відведеного тайм-ауту, наприклад 15 хвилин.

Підсистема «Звітність та документообіг» забезпечує збір аналітичних даних, розрахунок ключових показників ефективності (KPI) магазину подарунків та автоматизацію створення фінансових документів.

Гість не має доступу до жодних функцій цієї підсистеми. Покупець після успішної оплати замовлення може переглянути й завантажити електронний чек (Invoice) у форматі PDF з особистого кабінету та отримує згенерований чек на вказану електронну пошту як вкладення до листа з підтвердженням замовлення.

Адміністратор повинен мати можливість генерувати звіти про продажі за конкретний період, переглядати ключові фінансові метрики на головній панелі (загальний дохід, кількість успішних замовлень, середній чек), формувати рейтинг найпопулярніших товарів за обраний період для планування закупівель, експортувати аналітичні звіти у зовнішні формати (PDF для друку або CSV чи Excel для подальшого аналізу) та роздруковувати товарний чек або транспортну накладну для будь-якого замовлення під час пакування.

На рівні автоматизованих процесів система повинна автоматично формувати документ (Invoice) на основі заздалегідь підготовленого HTML-шаблону (DocumentTemplate) щоразу, коли замовлення переходить у статус «Оплачено» або «Виконано», розраховувати суму податків, якщо вони застосовуються, і відображати їх окремим рядком у згенерованому чеку, агрегувати дані з бази даних (підсистем замовлень і каталогу) за допомогою оптимізованих SQL-запитів для швидкої побудови аналітичних графіків без перевантаження сервера та забезпечувати безпечне зберігання згенерованих PDF-документів із обмеженням доступу, щоб завантажити чек міг лише власник замовлення або адміністратор.

Всі підсистеми магазину подарунків та зв'язки між ними відображено на рисунку 2.2.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						33

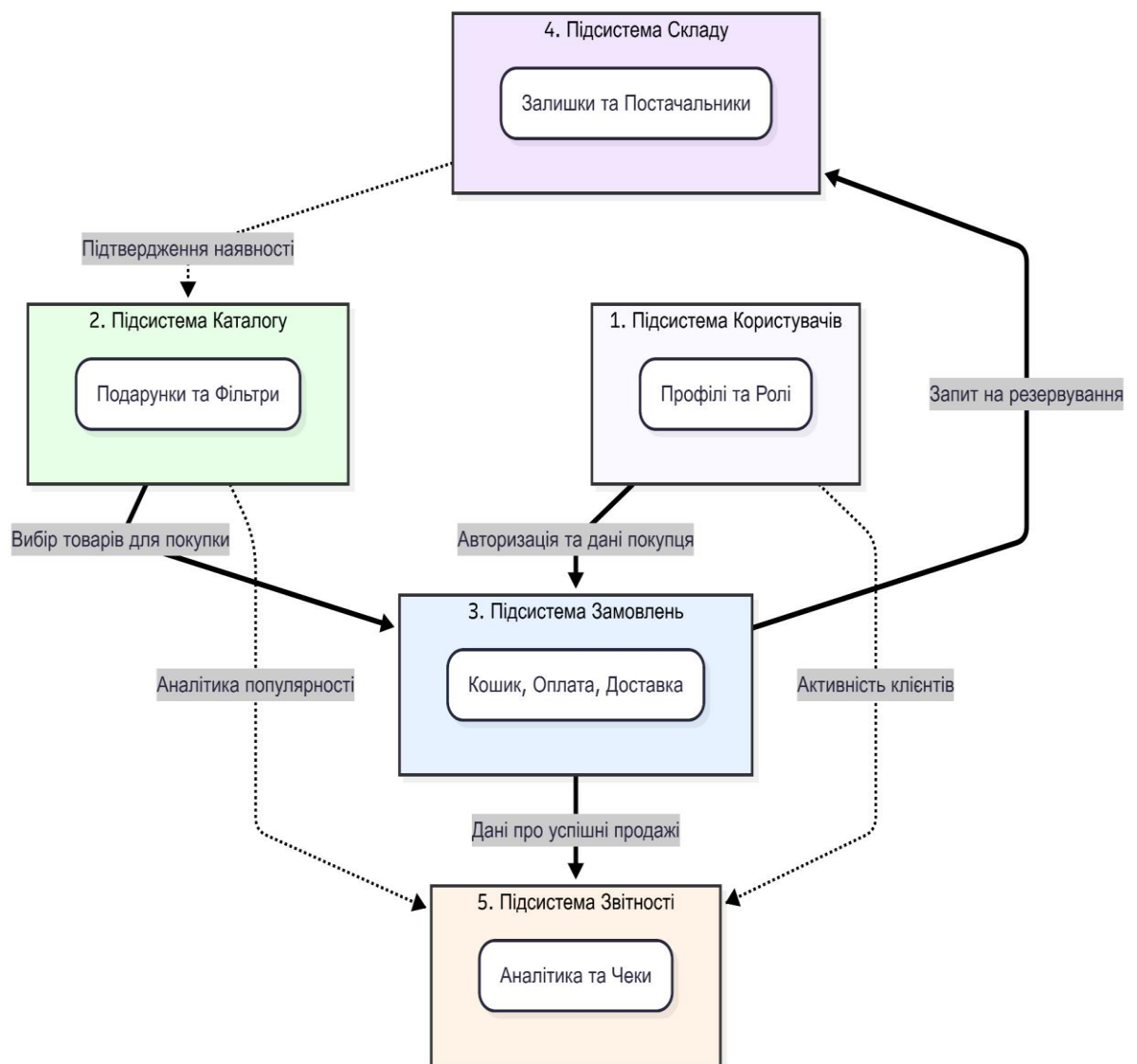


Рис. 2.2. Підсистеми магазину подарунків

Для деталізації внутрішньої структури програмного забезпечення, визначення об'єктної моделі та архітектурних зв'язків між компонентами системи було розроблено діаграму класів (Class Diagram) в нотації UML [8]. Оскільки архітектура розроблюваної вебплатформи має модульну структуру, логіку взаємодії клієнтської частини доцільно виділити в окремий архітектурний блок. Деталізовану структуру компонентів, типів даних, методів та атрибутів об'єктів користувацького інтерфейсу підсистеми користувача представлено на рис. 2.3.

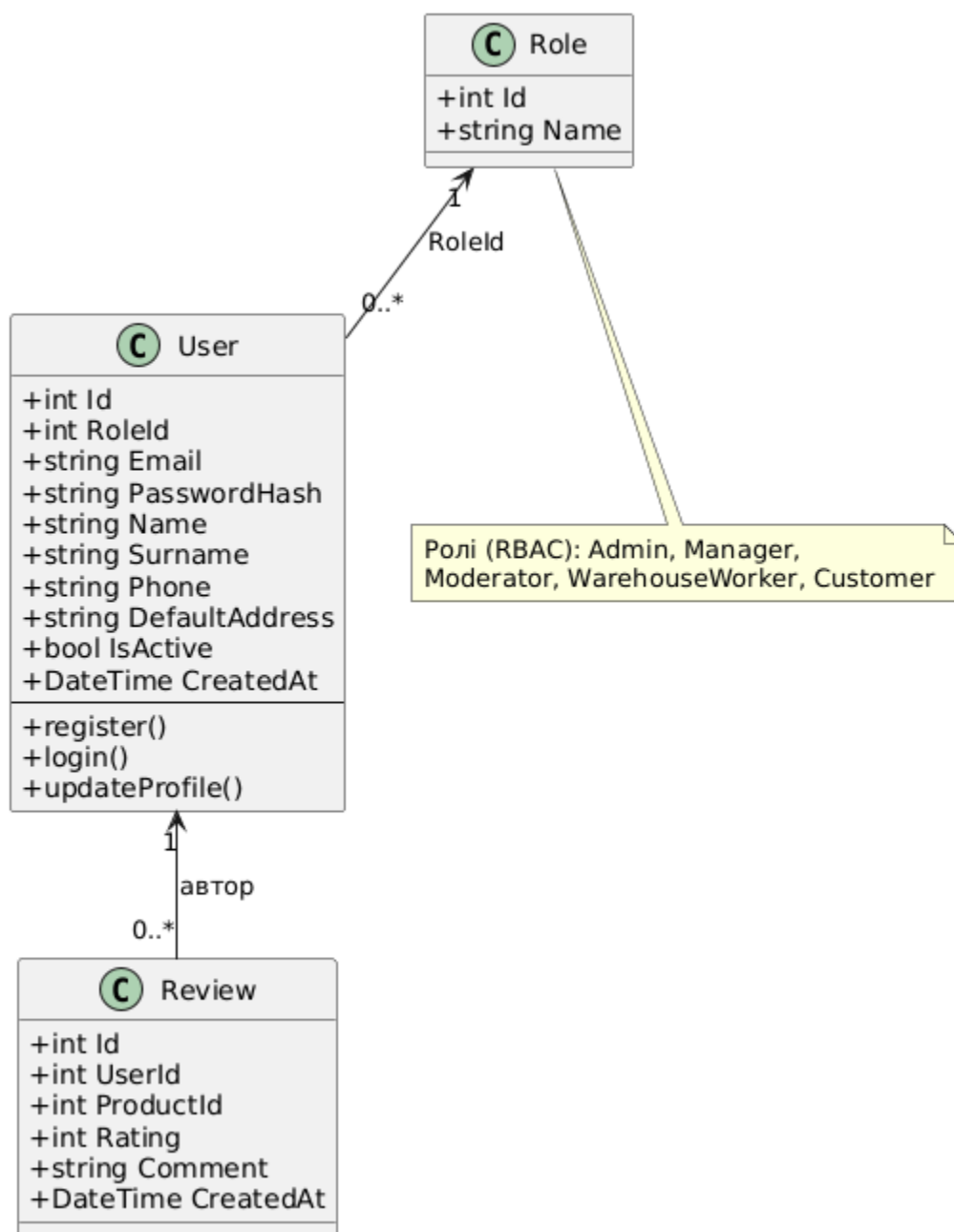


Рис. 2.3. Діаграма класів підсистеми користувача

Клас користувача (User) відповідає за дії, пов'язані з реєстрацією, входом і оновленням особистих даних клієнтів. За допомогою методу `register()` користувач може створити новий обліковий запис. Метод `login()` дає можливість пройти авторизацію, ввівши свої дані. Якщо потрібно змінити особисту інформацію, використовується метод `updateProfile()`

Клас ролей (Role) відповідає за класифікацію облікових записів та визначення їхніх прав доступу. Він дозволяє системі розрізняти типи користувачів: покупців (Customer), адміністраторів (Admin), менеджерів

(Manager), модераторів (Moderator) та працівників складу (WarehouseWorker), надаючи кожному відповідні повноваження.

Клас відгуків (Review) забезпечує збереження зворотного зв'язку від користувачів. Він містить інформацію про виставлену числову оцінку, залишений текстовий коментар та час створення запису. Взаємодія між цими сутностями побудована таким чином, що кожен користувач має певну роль у системі, а також може залишати відгуки, які автоматично прив'язуються до його облікового запису.

Наступним критично важливим компонентом об'єктної моделі вебплатформи є архітектурний модуль, який відповідає за збереження, фільтрацію, пошук та відображення товарної номенклатури. Для формалізації структури даних сувенірної продукції та визначення зв'язків між товарними категоріями, атрибутами й індивідуальними характеристиками подарунків було розроблено відповідну діаграму класів у нотації UML. Загальну структуру, методи та взаємозв'язки компонентів підсистеми продуктів представлено на рис. 2.4.

У системі передбачено кілька основних класів, кожен з яких відповідає за певну функціональність [12].

Клас товару (Product) відповідає за зберігання основної інформації про продукцію, включаючи її назву, детальний опис, вартість та статус доступності. За допомогою методу `updateStock()` можна оновлювати актуальну кількість одиниць певного товару, що залишилися на складі.

Клас категорії (Category) відповідає за систематизацію та класифікацію товарів, дозволяючи об'єднувати їх у відповідні групи та формувати ієрархічну структуру завдяки посиланню на батьківську категорію. Клас зображень товару (ProductImage) забезпечує збереження візуального контенту, містить посилання на файли та дозволяє визначати, яка саме фотографія є головною для відображення.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						36

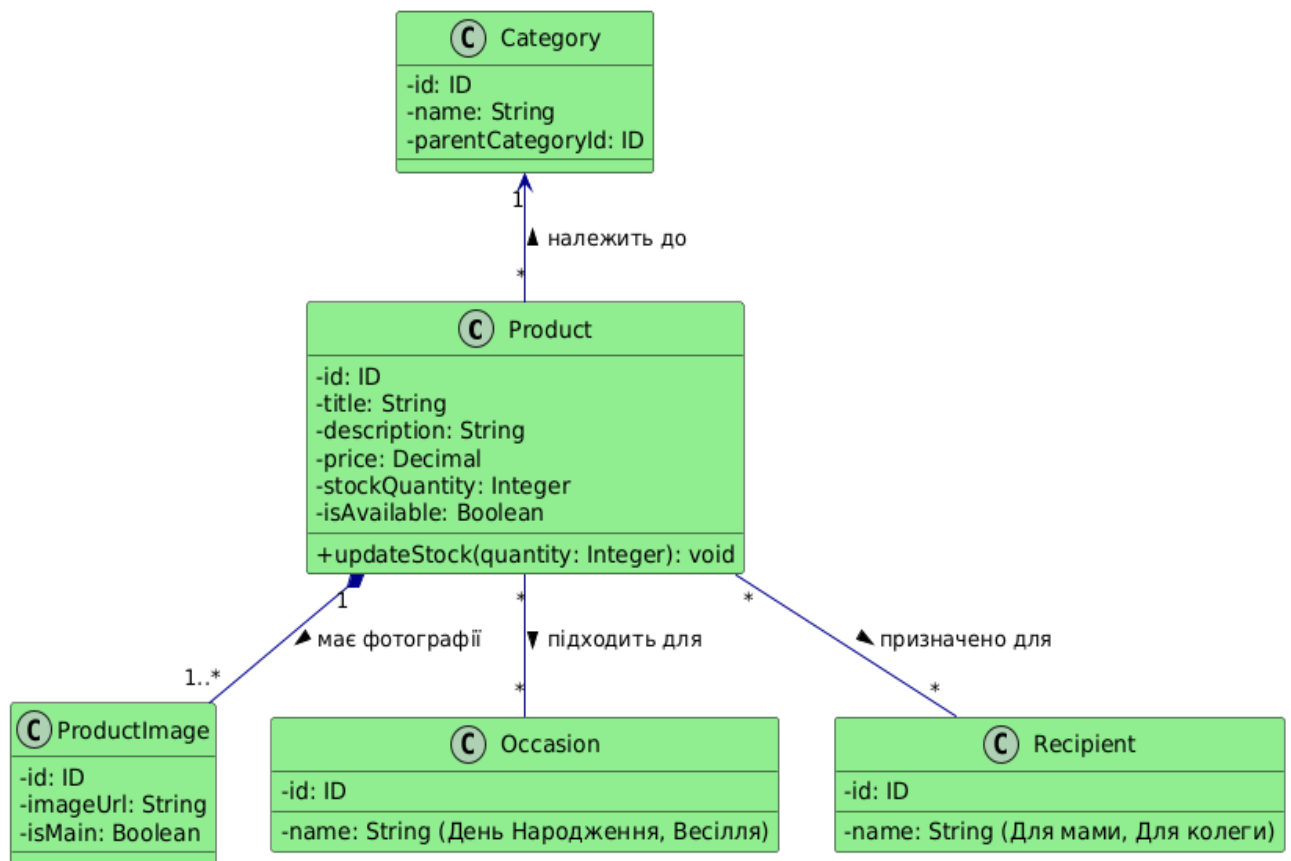


Рис. 2.4. Підсистема продуктів

Клас події (Occasion) містить інформацію про різноманітні свята чи заходи, такі як День народження або Весілля, для яких найкраще підходить конкретний товар. Клас одержувача (Recipient) відповідає за цільове призначення товару для певних категорій людей, наприклад, для мами чи для колеги. Взаємодія між сутностями побудована таким чином, що кожен товар обов'язково належить до певної категорії, має асоційовані зображення, а також може бути легко знайдений за фільтрами події або конкретного отримувача.

Наступним базовим модулем об'єктної моделі розроблюваної вебплатформи є архітектурний блок, що відповідає за реєстрацію, обробку та фіксацію транзакцій. Для формалізації процесів оформлення покупок, структурування фінансових метрик, інтеграції даних доставки та відстеження життєвого циклу кожної транзакції було спроектовано діаграму класів у нотації UML. Деталізовану структуру компонентів, атрибутів, методів та інтерфейсів підсистеми замовлень представлено на рис. 2.5.

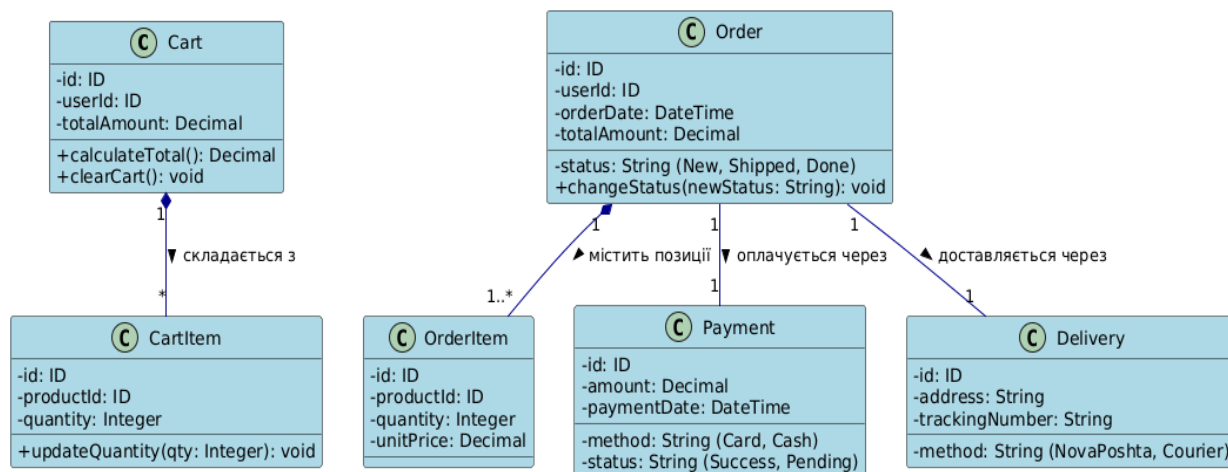


Рис. 2.5. Підсистема замовлень

У підсистемі замовлень передбачено кілька основних класів, кожен з яких відповідає за певну функціональність.

Клас замовлення (Order) відповідає за керування процесом оформлення покупки та життєвим циклом замовлення. За допомогою методу `createOrder()` створюється новий запис у базі даних при оформленні покупки. Метод `updateStatus()` дає можливість відстежувати та змінювати поточний стан виконання замовлення (наприклад, "очікує оплати", "в обробці" чи "відправлено"). Якщо клієнт вирішує відмовитися від покупки, використовується метод `cancelOrder()` для його скасування.

Клас елемента замовлення (OrderItem) деталізує інформацію про кожен окремий товар у сформованому замовленні, відповідаючи за збереження даних про замовлену кількість та зафіксовану на момент покупки ціну.

Клас обробки платежів (Payment) відповідає за фінансові операції та прив'язаний до конкретного замовлення. Метод `processPayment()` виконує обробку транзакції та підтверджує успішну оплату клієнтом. Клас доставки

(Delivery) керує інформацією про логістику, містить адресу призначення та дозволяє оновлювати статус відправлення за допомогою відповідних методів.

Додатково в підсистемі присутні клас кошика (Cart) та клас його елементів (CartItem), які відповідають за попереднє накопичення та тимчасове зберігання обраних товарів до моменту остаточного оформлення замовлення. Взаємодія між

цими сутностями гарантує повний та безперебійний цикл обслуговування: від додавання товару в кошик до його успішної оплати та доставки.

Специфіка функціонування сучасних платформ електронної комерції, орієнтованих на роботу з широким асортиментом продукції та сувенірів, вимагає наявності надійного інструментарію для оперативного контролю залишків. Для автоматизації процесів обліку товарних запасів, відстеження надходжень і списань, а також координації взаємодії між торговельним майданчиком та фізичними або віртуальними складськими приміщеннями було розроблено відповідну діаграму класів у нотації UML. Архітектурну структуру компонентів, атрибутів та методів підсистеми складу представлено на рис. 2.6.

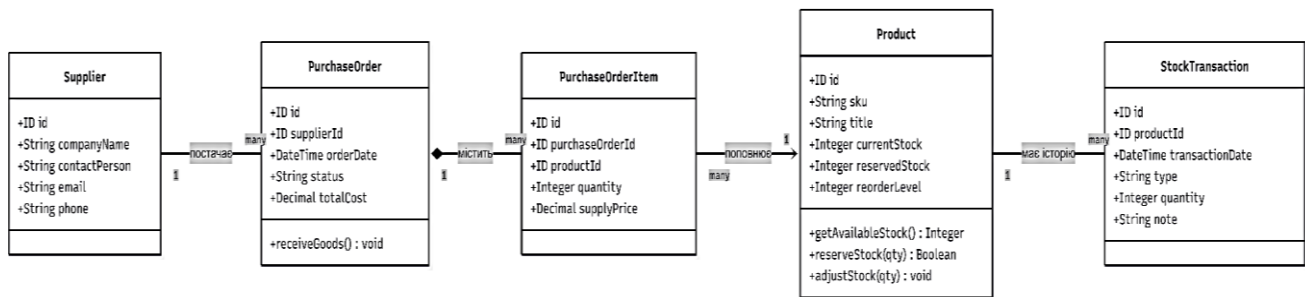


Рис. 2.6. Діаграма класів підсистеми складу

У підсистемі складу передбачено кілька основних класів, кожен з яких відповідає за певну функціональність.

Клас товару (Product) відповідає за облік продукції на складі, зберігаючи інформацію про її унікальний артикул (sku), назву, поточні залишки (currentStock) та зарезервовану кількість. За допомогою методу getAvailableStock() можна отримати інформацію про фактично доступну для вільного продажу кількість товару. Метод reserveStock() дає можливість тимчасово зарезервувати певну кількість одиниць, наприклад, при оформленні замовлення клієнтом. Якщо потрібно вручну скоригувати чи оновити загальну кількість товару на складі, використовується метод adjustStock().

Клас постачальника (Supplier) містить інформацію про компанії-партнери, які постачають продукцію. Він зберігає назву компанії, дані контактної особи та засоби зв'язку, забезпечуючи базу контрагентів для формування закупівель.

Клас замовлення на постачання (PurchaseOrder) відповідає за керування процесом закупівлі товарів у постачальників. Він фіксує дату створення, загальну вартість та статус виконання. Для підтвердження факту успішного надходження замовлених товарів на склад використовується метод receiveGoods(). Клас елемента закупівлі (PurchaseOrderItem) деталізує інформацію, вказуючи конкретний товар, його кількість та закупівельну ціну в межах одного замовлення постачальнику.

Клас складських транзакцій (StockTransaction) відповідає за детальну фіксацію історії руху товарів. Він зберігає записи про будь-які зміни кількості на складі, тип операції (наприклад, надходження, списання) та точний час транзакції. Взаємодія між цими сутностями забезпечує повний та прозорий контроль над запасами: від закупівлі у постачальника до точного відстеження кожної зміни залишків продукції.

Для забезпечення ефективного стратегічного управління, оцінки комерційної успішності платформи та моніторингу технічних показників системи виникає необхідність впровадження аналітичного інструментарію. З метою формалізації процесів збору, агрегації та візуалізації бізнес-метрики (таких як обсяги продажів, конверсія, активність користувачів та стан складських запасів) було спроектовано діаграму класів у нотації UML. Організаційну структуру компонентів, атрибутів, методів та інтерфейсів підсистеми звітності представлено на рис. 2.7.

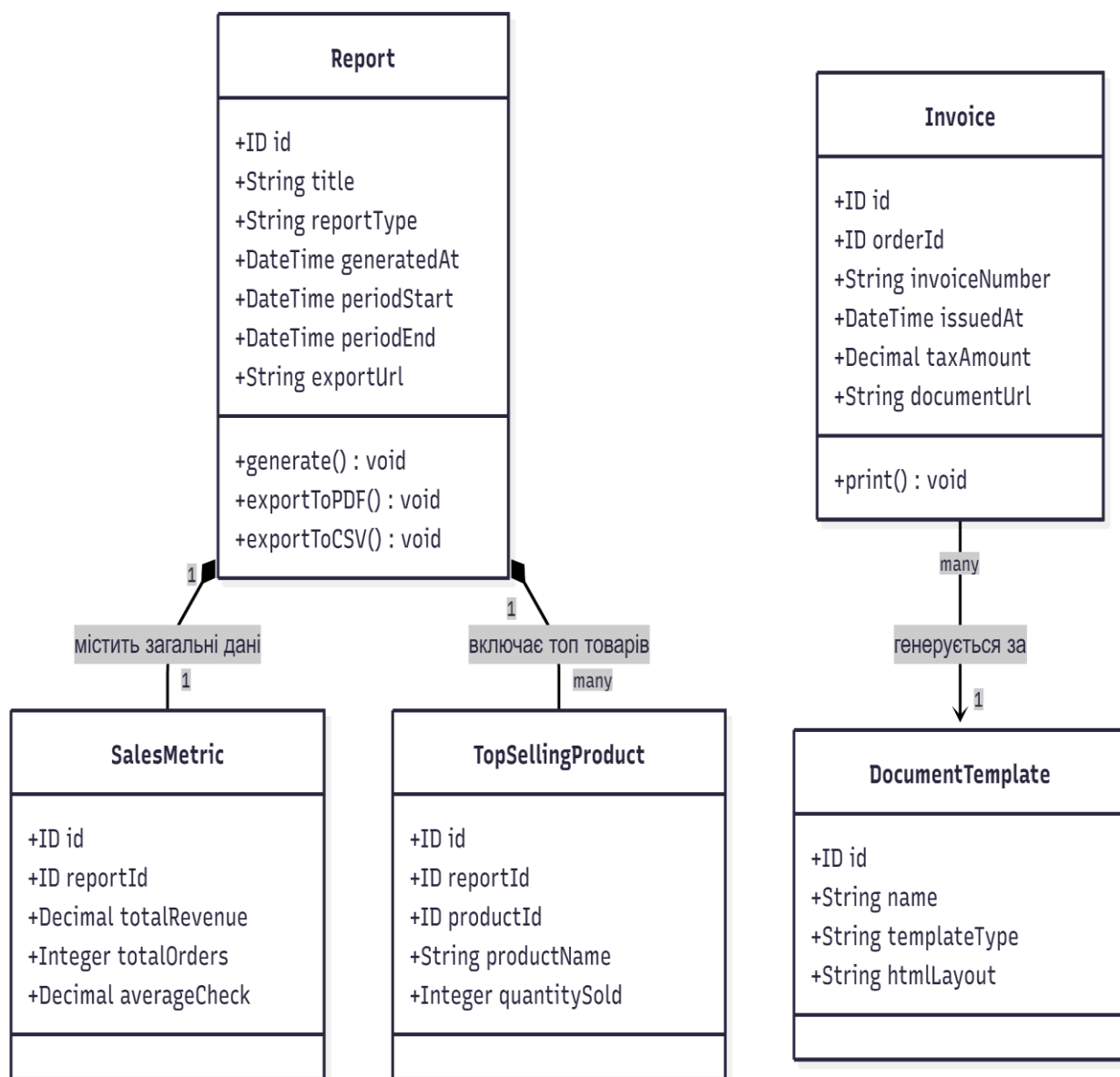


Рис. 2.7. Діаграма класів підсистеми звітності

У підсистемі звітності передбачено кілька основних класів, кожен з яких відповідає за певну функціональність.

Клас звіту (Report) відповідає за формування та збереження аналітичної інформації за визначений проміжок часу, фіксуючи дату початку (periodStart) та закінчення (periodEnd) періоду. За допомогою методу generate() система створює новий звіт на основі зібраних даних. Для подальшого використання чи поширення інформації передбачено методи експорту: exportToPDF() дозволяє зберегти звіт у форматі PDF, а exportToCSV() – вивантажити його у вигляді таблиці CSV.

Клас показників продажів (SalesMetric) містить загальні зведені дані для конкретного звіту, такі як загальний дохід (totalRevenue), загальна кількість замовлень (totalOrders) та середній чек (averageCheck). Клас найпопулярніших товарів (TopSellingProduct) відповідає за деталізацію звіту, включаючи інформацію про конкретні продукти, їхні назви та кількість проданих одиниць (quantitySold). Взаємодія між цими сутностями побудована так, що один звіт містить єдиний набір загальних фінансових метрик, але може включати перелік із багатьох найпопулярніших товарів.

Клас рахунку-фактури (Invoice) відповідає за генерацію фінансової документації. Він зберігає інформацію про унікальний номер документа (invoiceNumber), прив'язку до замовлення (orderId), дату виставлення та суму нарахованого податку (taxAmount). За допомогою методу print() користувач може роздрукувати готовий рахунок або отримати його за відповідним посиланням (documentUrl).

Клас шаблону документа (DocumentTemplate) забезпечує стандартизацію зовнішнього вигляду згенерованих файлів. Він містить назву шаблону, його тип та безпосередньо HTML-розмітку (htmlLayout). Взаємодія показує, що безліч рахунків-фактур можуть генеруватися на основі одного обраного шаблону, забезпечуючи єдиний корпоративний стиль документації.

2.2. Розробка бази даних системи

Для забезпечення надійного зберігання, швидкої обробки даних та підтримки різноманітних зв'язків між об'єктами, розроблювана система використовує реляційну базу даних MS SQL Server. Вона є потужною системою керування базами даних і забезпечує ефективну взаємодію між компонентами бекенду та клієнтською частиною [29, 30].

Структура бази даних була ретельно продумана і спроектована з урахуванням усіх аспектів діяльності магазину подарунків. Особливу увагу приділено гнучкості та розширюваності, що дозволяє безперешкодно

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						42

масштабувати функціонал у майбутньому без необхідності повної перебудови основи системи. Нижче наведена логічна схема бази даних (рис. 2.8).

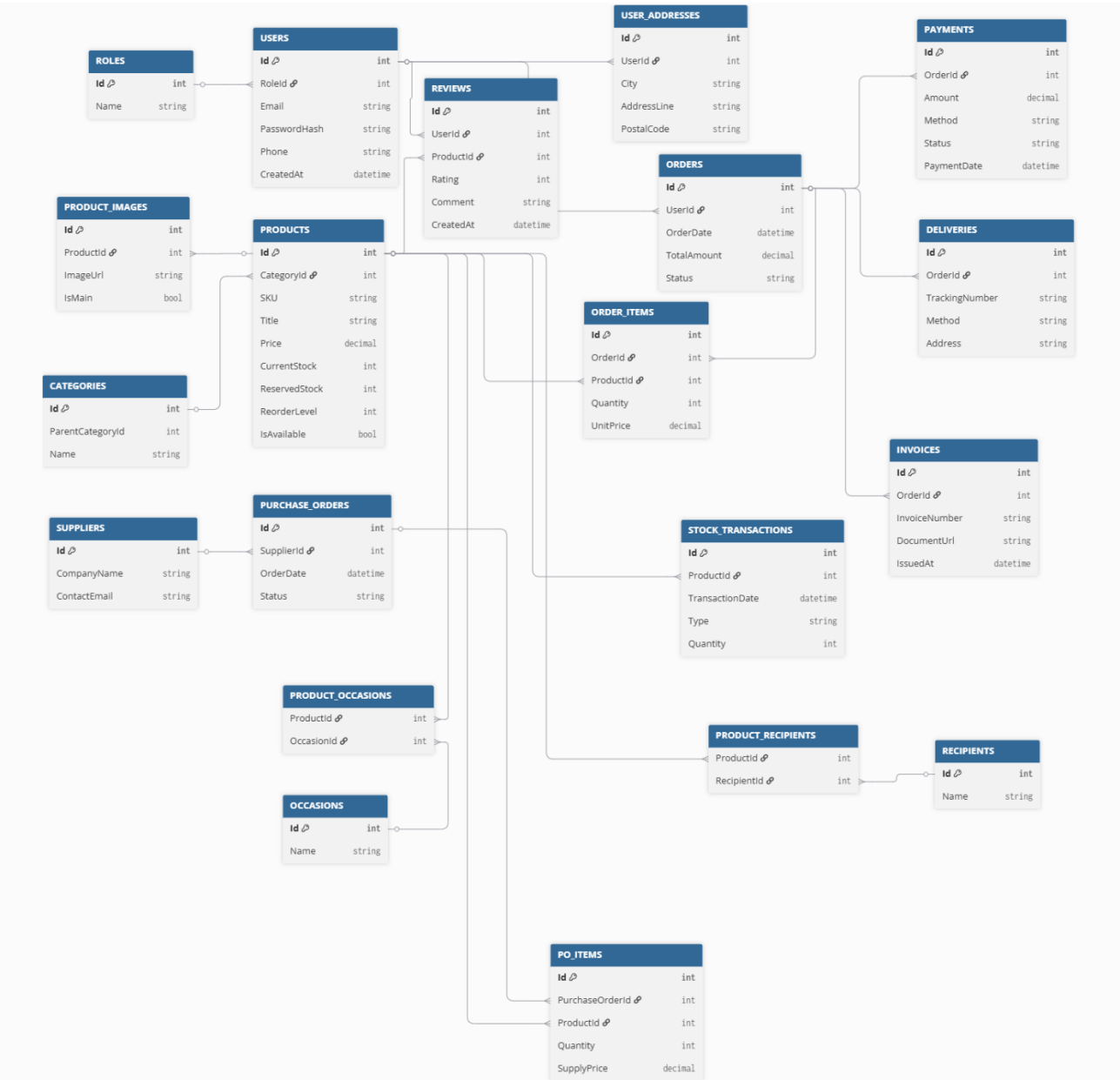


Рис. 2.8. Логічна схема бази даних

Однією з найважливіших складових бази даних є таблиця Users (табл. 2.1). Саме вона зберігає інформацію про клієнтів, які можуть переглядати товари, оформлювати замовлення, проходити реєстрацію та авторизацію. Окрім основних ідентифікаційних полів, тут також зберігається контактний телефон, а для безпеки всі паролі зберігаються в хешованому вигляді.

Таблиця 2.1

Структура таблиці Users

Поле	Тип	Опис
Id	INT PRIMARY KEY	Ідентифікатор користувача
RoleId	INT FOREIGN KEY	Ідентифікатор ролі доступу
Email	VARCHAR(255) NOT NULL	Електронна пошта користувача
PasswordHash	VARCHAR(255) NOT NULL	Хешований пароль
Phone	VARCHAR(20) NULL	Контактний телефон
CreatedAt	DATETIME	Дата створення профілю

Таблиця Products (табл. 2.2) є центральною складовою бази даних будь-якого електронного магазину, адже саме вона містить найважливіше – перелік усіх товарів, які потенційно можуть бути придбані користувачами.

Ця таблиця виступає своєрідним «вітринним каталогом» системи, де зберігаються всі ключові атрибути, що дозволяють не лише коректно відображати товар у користувацькому інтерфейсі, а й оперативно реагувати на зміну попиту, залишків чи цін. Крім того, грамотне проєктування структури цієї сутності є критично важливим для забезпечення високої продуктивності бази даних під час складних пошукових запитів. Використання зовнішніх ключів для зв'язку з довідниками (категорії, виробники) забезпечує необхідний рівень нормалізації, що гарантує транзакційну цілісність інформації та дозволяє безперешкодно масштабувати архітектуру в майбутньому.

Таблиця 2.2

Структура таблиці Products

Поле	Тип	Опис
Id	INT PRIMARY KEY	Ідентифікатор товару
CategoryId	INT FOREIGN KEY	Посилання на категорію
SKU	VARCHAR(100)	Артикул товару (штрихкод)
Title	VARCHAR(255) NOT NULL	Назва подарунка
Price	DECIMAL(10, 2)	Вартість товару
CurrentStock	INT	Фактичний залишок на складі
ReservedStock	INT	Зарезервована кількість під час оплати
IsAvailable	BIT	Статус доступності (1 – так, 0 – ні)

Коли користувач оформлює покупку, усі дані автоматично зберігаються в таблиці Orders, що виступає центральним джерелом інформації про всі замовлення на платформі. У цій таблиці фіксуються не лише основні атрибути замовлення, як-от посилання на покупця чи загальна сума, а й зберігається

актуальний статус транзакції. Це дозволяє адміністраторам легко та оперативно відслідковувати успішність покупок, обробляти нові замовлення або виявляти потенційні проблеми.

Таблиця 2.3

Структура таблиці Orders

Поле	Тип	Опис
Id	INT PRIMARY KEY	Ідентифікатор замовлення
UserId	INT FOREIGN KEY	Посилання на клієнта
OrderDate	DATETIME	Дата та час оформлення
TotalAmount	DECIMAL(10, 2)	Загальна сума до сплати
Status	VARCHAR(50)	Поточний статус (Нове, В обробці)

Окрім наведених, база даних містить низку інших взаємопов'язаних сутностей, відображених на логічній схемі (рис. 2.8). Таблиця Categories забезпечує ієрархічну класифікацію товарів завдяки посиланню на батьківську категорію, а таблиця ProductImages зберігає набір зображень для кожного товару з ознакою головного. Зв'язок товарів із приводами та одержувачами реалізовано за схемою «багато-до-багатьох» через проміжні таблиці ProductOccasions і ProductRecipients, що дає змогу одному товару належати до кількох приводів одночасно, наприклад «День народження» та «Новий рік». Деталізацію кожного замовлення зберігає таблиця OrderItems, у якій фіксуються замовлена кількість і ціна товару на момент покупки, що убезпечує історичні дані від подальших змін цін у каталозі.

Складську підсистему обслуговують таблиці Suppliers, PurchaseOrders і StockTransactions, які зберігають відомості про постачальників, замовлення на постачання та повну історію руху товарних залишків. Фінансову та аналітичну складову забезпечують таблиці Invoices, Payments і Deliveries, пов'язані із замовленнями, а також таблиця Reviews для зберігання відгуків і рейтингів покупців. Розмежування прав доступу реалізовано через таблицю Roles, на яку посилається таблиця Users, а збережені адреси доставки клієнтів зберігає таблиця UserAddresses.

Проектування бази даних виконано з дотриманням принципів нормалізації до третьої нормальної форми (3НФ), що мінімізує надмірність і дублювання

даних. Цілісність забезпечується системою первинних та зовнішніх ключів: кожна таблиця має унікальний первинний ключ (Id), а зв'язки між сутностями підтримуються зовнішніми ключами з обмеженнями посилальної цілісності. Для полів, що потребують унікальності (електронна пошта користувача, артикул товару SKU), встановлено обмеження UNIQUE, а для обов'язкових атрибутів – обмеження NOT NULL.

Особливу увагу приділено узгодженості даних під час критичних операцій. Створення замовлення, додавання його позицій та оновлення складських залишків виконуються в межах єдиної транзакції відповідно до вимог ACID: у разі помилки на будь-якому етапі всі зміни скасовуються, що унеможливорює появу неповних замовлень або некоректних значень залишків. Для підвищення продуктивності вибірок на стовпцях, які найчастіше беруть участь у фільтрації та з'єднанні таблиць (зовнішні ключі, артикул товару, електронна пошта), передбачено індекси, що пришвидшує виконання типових запитів каталогу та обробки замовлень.

Фізичну реалізацію бази даних здійснено засобами об'єктно-реляційного відображення (ORM) Entity Framework Core за підходом Code-First. Кожну сутність предметної області описано у вигляді класу мовою C#, а контекст даних ApplicationDbContext визначає набори сутностей (DbSet) і правила їх відображення на таблиці. Створення та подальше оновлення структури бази даних виконуються через механізм міграцій (Migrations), який дає змогу версіонувати зміни схеми й синхронізувати її з моделлю даних без ручного написання SQL-скриптів. Тонке налаштування зв'язків, типів полів та обмежень реалізовано за допомогою Fluent API та атрибутів (Data Annotations).

Зв'язки між сутностями побудовано відповідно до правил реляційної моделі та бізнес-логіки магазину. Відношення «один-до-багатьох» діє між таблицями Users і Orders (один користувач може мати багато замовлень), а також між Categories і Products, Orders і OrderItems, Products і ProductImages, Suppliers і PurchaseOrders. Відношення «один-до-одного» застосовано між таблицею Orders та сутностями Payment, Delivery й Invoice, оскільки кожному замовленню

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						46

відповідає рівно один платіж, одна доставка та один рахунок-фактура. Ієрархію категорій довільної глибини реалізовано самопосиланням таблиці Categories через поле ParentCategoryId, а відношення «багато-до-багатьох» між товарами, приводами та одержувачами – через проміжні таблиці.

Під час проєктування окрему увагу приділено обґрунтованому вибору типів даних. Для ідентифікаторів застосовано цілочисельний тип INT з автоінкрементом, для текстових полів (назви, описи, адреси електронної пошти) – тип VARCHAR відповідної довжини, а для грошових значень – тип DECIMAL(10, 2), який, на відміну від чисел із рухомою комою, гарантує точне зберігання сум без похибок округлення. Часові позначки зберігаються у форматі DATETIME, а логічні ознаки доступності чи активності – у компактному типі BIT.

Цілісність значень на рівні системи керування базами даних додатково підкріплено перевірочними обмеженнями (CHECK): ціна товару та кількість складських залишків не можуть набувати від’ємних значень, а виставлена у відгуку оцінка обмежена діапазоном від одного до п’яти балів. Для полів, що фіксують дату створення запису, встановлено значення за замовчуванням (DEFAULT), яке автоматично проставляє поточну дату й час, позбавляючи застосунок потреби формувати цю позначку вручну.

Важливим аспектом проєктування є коректне опрацювання конкурентного доступу до складських залишків. Щоб запобігти продажу однієї й тієї самої останньої одиниці товару двом покупцям одночасно, реалізовано механізм резервування: під час переходу до оплати потрібна кількість переноситься з поля CurrentStock до поля ReservedStock у межах єдиної транзакції. Для виявлення конфліктних одночасних змін застосовано оптимістичне блокування, яке відстежує версію рядка та відхиляє неузгоджені оновлення, що особливо актуально під час пікових святкових навантажень.

Для забезпечення працездатності системи одразу після розгортання передбачено початкове наповнення довідкових таблиць (seeding): у таблицю Roles автоматично додаються базові ролі системи та початковий обліковий запис

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	47
		Громський О.О.				

адміністратора, що дає змогу розпочати керування магазином без ручного налаштування бази даних. Спроектowana таким чином структура бази даних поєднує сувору нормалізацію, цілісність і високу продуктивність та залишається гнучкою до подальшого розширення функціональності магазину подарунків.

2.3. Проєктування алгоритмів, архітектури RESTful API та бізнес-процесів

Для того щоб інтернет-магазин працював плавно, логічно й при цьому залишався інтуїтивно зрозумілим для користувача, у процесі розробки було реалізовано кілька ключових механізмів, які забезпечують коректну взаємодію між елементами системи. Ці механізми дозволяють користувачам не тільки зручно шукати товари, оформлювати замовлення, але й дають адміністрації змогу керувати платформою, а також регулювати процеси, пов'язані з відгуками.

Весь внутрішній функціонал реалізовано у вигляді окремих логічних блоків, які відповідають за певну операцію – фільтрацію, створення замовлення, перевірку залишків на складі, а також оновлення статусів чи взаємодію з користувачами.

Один із найважливіших процесів, що реалізований у системі, – це механізм фільтрації товарів. Коли користувач заходить на сайт і починає шукати ідеальний подарунок, йому хочеться якнайшвидше звузити вибір і побачити тільки ті позиції, які його дійсно цікавлять. Щоб це забезпечити, система дозволяє відбирати товари за цілою низкою специфічних параметрів: категорією, приводом свята (наприклад, "День народження"), отримувачем ("Для мами", "Колесі"), вартістю або наявністю.

Користувач вибирає необхідні опції, і система формує запит до бази даних MS SQL, який повертає саме ті результати, які відповідають зазначеним умовам. Усе це відбувається у фоновому режимі, миттєво і без оновлення сторінки, що забезпечує приємний досвід використання. Такий підхід дозволяє значно покращити ефективність пошуку, а також зменшує навантаження на інтерфейс.

Ще одним важливим процесом, який реалізовано у межах роботи магазину, є створення замовлення. Цей процес хоч і виглядає зовні доволі простим для звичайного користувача – додати товар у кошик і натиснути «Оформити» – але за ним стоїть чітко прописана транзакційна логіка бекенду на ASP.NET Core, яка відповідає за фіксацію замовлення в системі. Коли покупець вирішує замовити певні товари, система формує запис у базі даних, в якому зберігаються всі основні дані: хто зробив замовлення, які саме товари було обрано, їх кількість, загальна сума та час створення замовлення.

На відміну від спрощених систем, наша платформа передбачає механізм динамічного резервування товару: під час переходу до оплати кількість товару з поля CurrentStock тимчасово переноситься у ReservedStock. Це гарантує, що два користувачі не зможуть одночасно придбати останній екземпляр унікального подарунка. Після успішної оплати замовлення йому присвоюється відповідний статус, а товари остаточно списуються зі складу. Адміністратор або менеджер магазину надалі бачить усі активні замовлення, переглядає деталі й приймає рішення про їх відправлення.

Не менш важливою частиною є перевірка відгуків. Щоб забезпечити якість і чесність на платформі, кожен відгук, який залишає користувач, проходить перевірку модератором. Це зроблено для того, щоб уникнути спаму, неправдивої інформації або нецензурної лексики. Як тільки модератор бачить новий відгук, він може прийняти рішення – схвалити його, якщо все відповідає правилам, або відхилити, якщо в ньому є порушення.

Програмну реалізацію описаних процесів на клієнтській частині наведено в додатку. Ключову операцію панелі адміністратора ілюструє метод `handleStatusChange`, який оновлює статус замовлення через серверний сервіс і повідомляє про результат (див. лістинг 2.1). Оформлення замовлення на стороні клієнта виконує метод `submitOrder`: він перевіряє коректність введених даних, формує об'єкт замовлення з товарів кошика та обраних способів доставки й оплати і надсилає його на сервер (див. лістинг 2.2). Логіку панелі керування складом, яка під час відкриття сторінки завантажує звіт про потребу в поповненні

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						49

запасів, наведено в лістингу 2.3. Конфігурацію маршрутизації застосунку з переліком маршрутів та обробником `beforeEach`, що перевіряє автентифікацію й роль користувача, показано в лістингу 2.4.

Для моделювання динамічного аспекту архітектури програмного забезпечення, візуалізації взаємодії між об'єктами системи у часі та деталізації процесу обміну повідомленнями під час виконання ключових бізнес-сценаріїв було розроблено діаграму послідовності (Sequence Diagram) в нотації UML [9]. Цей інструмент проєктування дозволяє простежити хронологію виклику методів та передачі даних між клієнтським інтерфейсом, керуючими контролерами, сервісами бізнес-логіки та рівнем бази даних. Загальний вигляд розробленої діаграми послідовності вебплатформи магазину подарунків представлено на рис. 2.9.

Усі ці процеси працюють в рамках єдиної архітектури системи, де кожен компонент має свою чітку роль. На рисунку нижче зображено один із таких процесів – модерацію відгуку. На діаграмі показано, як система реагує на дії модератора: від моменту, коли він відкриває список нових відгуків, до моменту, коли оновлюється статус коментаря у базі даних.

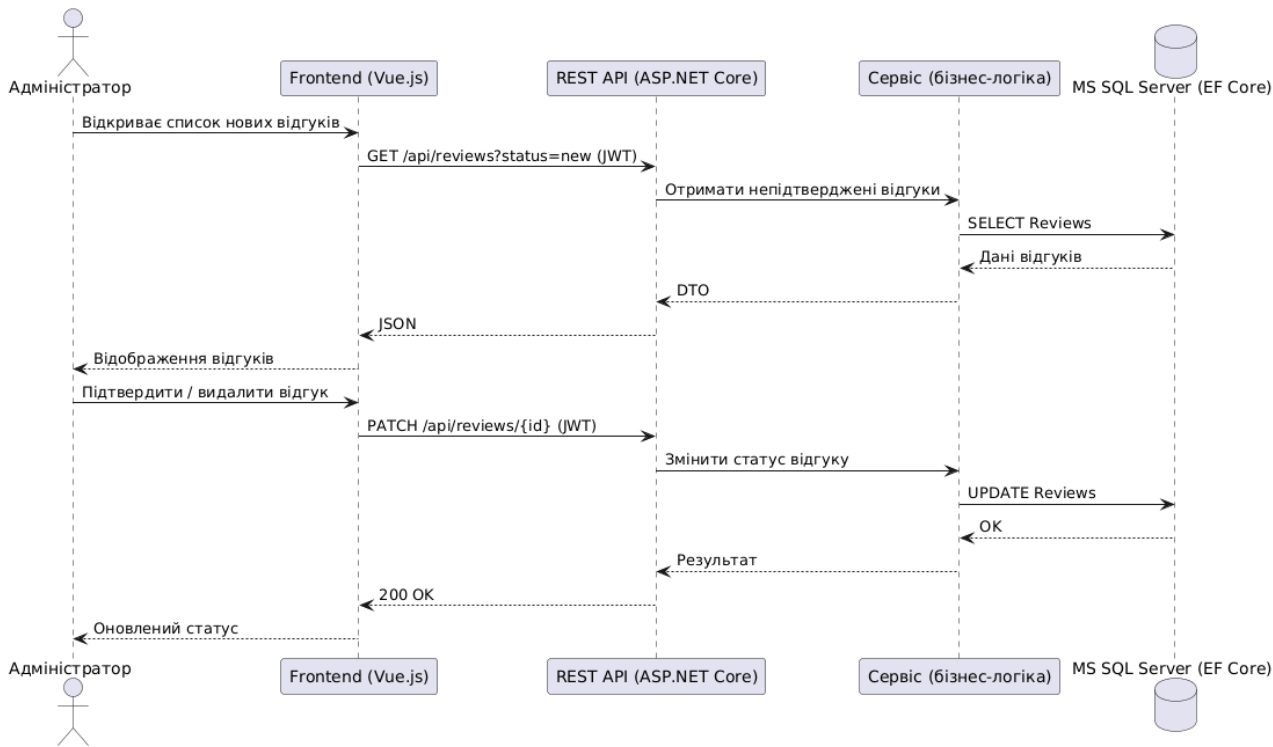


Рис. 2.9. Діаграма послідовності (модерація відгуку)

Наведена діаграма послідовності демонструє порядок обміну повідомленнями між клієнтським інтерфейсом, керуючими контролерами, сервісами бізнес-логіки та базою даних на прикладі модерації відгуку: від відкриття списку нових відгуків до оновлення статусу коментаря у базі даних.

Оскільки обрана клієнт-серверна архітектура передбачає чіткий поділ між клієнтським додатком на Vue.js та сервером на ASP.NET Core, взаємодія між ними реалізована за допомогою RESTful API (Representational State Transfer). Цей архітектурний стиль дозволяє стандартизувати запити та зробити систему легко масштабованою [33].

Усі обміни даними відбуваються у форматі JSON (JavaScript Object Notation). Для кожної сутності в системі спроектовано відповідні контролери (Controllers), які обробляють HTTP-запити: GET (отримання даних), POST (створення), PUT/PATCH (оновлення) та DELETE (видалення).

Для ілюстрації роботи серверної частини в таблиці 2.4 наведено опис основних кінцевих точок (endpoints) для підсистем «Каталог» та «Замовлення».

Успішні запити повертають HTTP-статуси 200 (OK) або 201 (Created). У разі помилок валідації система повертає код 400 (Bad Request), а при спробі несанкціонованого доступу до адміністративних маршрутів – 401 (Unauthorized) або 403 (Forbidden).

Таблиця 2.4

Специфікація основних кінцевих точок RESTful API

HTTP-метод	Маршрут (Endpoint)	Опис дії	Права доступу
GET	/api/products	Отримання списку товарів із застосуванням параметрів пагінації та фільтрації (за категорією, ціною)	Всі користувачі
GET	/api/products/{id}	Отримання детальної інформації про конкретний товар за його ідентифікатором	Всі користувачі
POST	/api/products	Створення нового товару в базі даних	Лише Адміністратор
GET	/api/categories	Отримання ієрархічного дерева категорій товарів	Всі користувачі
POST	/api/orders	Створення нового замовлення на основі вмісту кошика (Checkout)	Авторизований клієнт

GET	/api/orders/my	Отримання історії замовлень для поточного авторизованого користувача	Авторизований клієнт
PATCH	/api/orders/{id}/status	Оновлення статусу замовлення по мірі його виконання	Лише Адміністратор
POST	/api/auth/login	Авторизація користувача, перевірка пароля та генерація JWT-токена [21]	

Для кращого розуміння динаміки роботи вебплатформи було спроектовано додаткові UML-діаграми, які описують поведінку системи під час оформлення та обробки замовлення.

Життєвий цикл замовлення в інтернет-магазині «Подарунковий магазин» проходить через низку станів, переходи між якими ініціюють дії клієнта, адміністратора або автоматичні тригери системи. У стані «Нове» (New) замовлення щойно створене клієнтом через кошик, і товар тимчасово резервується на складі. Стан «В обробці» (Processing) настає після підтвердження та успішної оплати, коли адміністратор отримує сповіщення й починає пакування подарунка. У стані «Відправлено» (Shipped) товар передано логістичній компанії, а клієнтові надсилають трек-номер для відстеження. Стан «Виконано» (Completed) означає, що клієнт отримав замовлення і транзакцію успішно закрито. Якщо клієнт відмовляється від посилки при отриманні на пошті, замовлення переходить у стан «Повернено» (Returned). Стан «Скасовано» (Cancelled) може настати зі стану «Нове» у разі відмови клієнта чи відхилення замовлення системою або зі стану «В обробці», якщо адміністратор скасовує його через відсутність товару.

Для детального аналізу поведінкової логіки системи та моделювання життєвого циклу ключових транзакційних сутностей було розроблено діаграму станів (State Machine Diagram) в нотації UML [9].

Цей тип моделювання є критично важливим для підсистеми замовлень, оскільки дозволяє формалізувати всі можливі внутрішні стани, у яких може перебувати об'єкт замовлення з моменту його ініціалізації клієнтом до моменту успішного архіваційного завершення або скасування. Хронологію переходів, тригери подій та захисні умови для процесу обробки покупок представлено на рис. 2.10.

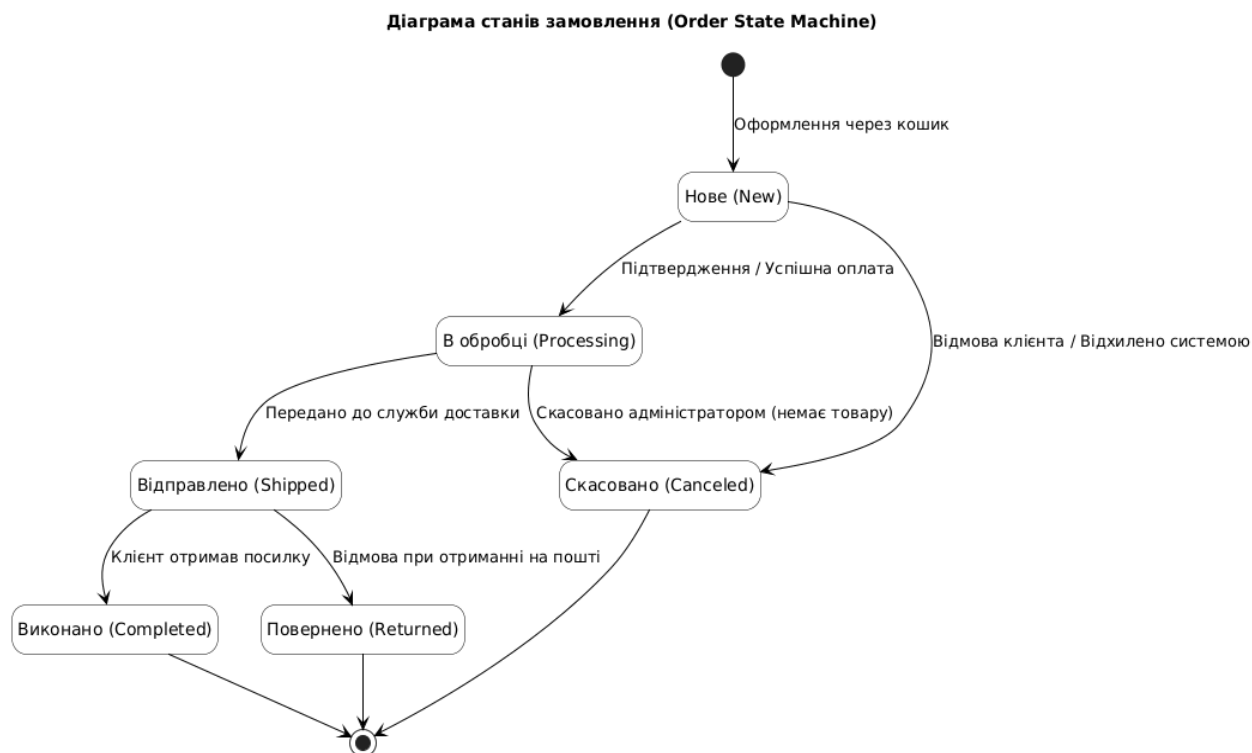


Рис. 2.10. Діаграма станів оформлення замовлення на вебплатформі магазину подарунків

Для детального аналізу алгоритмічної складової бізнес-процесів вебплатформи та візуалізації покрокового процедурного інструменту взаємодії користувача з додатком було розроблено діаграму активності (Activity Diagram) в нотації UML [9].

Цей поведінковий тип моделювання дозволяє відобразити паралельні та послідовні потоки дій, умови розгалуження логіки системи, а також точки прийняття рішень під час взаємодії покупця з інтерфейсом Checkout. Покрокову послідовність операцій та керуючі структури процесу оформлення покупок представлено на рис. 2.11.

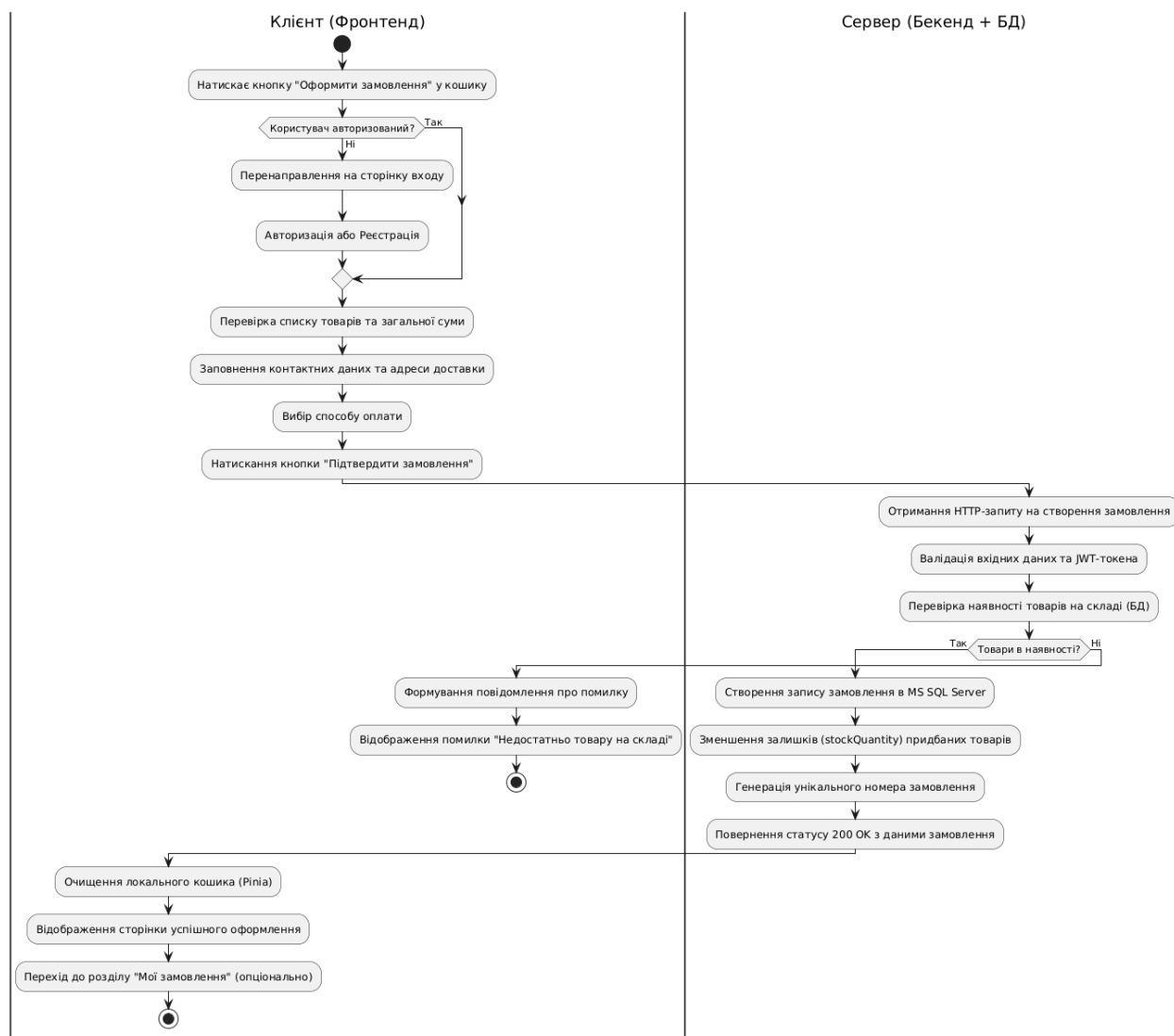


Рис. 2.11. Діаграма активності (оформлення замовлення)

Процес оформлення покупки вимагає суворої послідовності кроків для забезпечення консистентності даних, і діаграма активності моделює цей потік. Процес починається з переходу користувача до кошика, після чого система перевіряє статус авторизації: якщо клієнт є гостем, йому пропонують авторизуватися або пройти швидку реєстрацію. Далі система перевіряє актуальні залишки (Available Stock) для кожного товару в кошику, і якщо хоча б одного товару недостатньо, користувач отримує попередження та пропозицію змінити кількість. Після успішної перевірки клієнт обирає спосіб доставки та оплати, а на фінальному етапі система створює запис у таблиці Orders, ініціює фінансову транзакцію та генерує електронний чек, який автоматично надсилається на email покупця.

Побудована діаграма активності наочно відображає послідовність дій та точки прийняття рішень під час оформлення замовлення, що дає змогу простежити логіку взаємодії покупця із системою на кожному кроці.

Висновки до другого розділу

У другому розділі було проведено детальне проектування архітектури та обґрунтування вибору технологічних рішень для інформаційної системи «Подарунковий магазин». На основі проведеної роботи можна сформулювати такі висновки:

Сформовано повний перелік вимог до системи, що включає функціональні можливості (керування каталогом, кошиком, замовленнями та автентифікацію) та нефункціональні вимоги (продуктивність, безпека даних та масштабованість). Це дозволило чітко окреслити межі розробки та критерії успішності проєкту.

Обґрунтовано вибір клієнт-серверної архітектури на базі Single Page Application (SPA). Такий підхід забезпечує високу швидкість взаємодії користувача з інтерфейсом та дозволяє розділити логіку представлення (фронтенд) від бізнес-логіки обробки даних (бекенд).

Розроблено структуру бази даних, яка базується на реляційній моделі. Спроектowana схема даних у MS SQL Server забезпечує цілісність та ефективний зв'язок між основними сутностями: товарами, категоріями, приводами (occasions), користувачами та замовленнями. Особливу увагу приділено оптимізації зв'язків для швидкої фільтрації товарів за різними критеріями.

Визначено оптимальний технологічний стек.

Для серверної частини обрано платформу .NET Core (C#) через її надійність, підтримку асинхронності та вбудовані механізми безпеки. Для клієнтської частини обрано Vue.js у поєднанні зі стейт-менеджером Pinia, що дозволяє ефективно керувати станом кошика та забезпечує реактивність інтерфейсу.

Спроектowana систему безпеки на основі стандарту JWT (JSON Web Token). Це дозволяє реалізувати надійну автентифікацію без зберігання станів на

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						55

сервері та забезпечує гнучке розмежування прав доступу на основі моделі ролей користувачів (RBAC).

Результати проєктування, отримані у цьому розділі, стали фундаментом для програмної реалізації та тестування системи, що детально описані у наступному етапі роботи.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						56

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ

3.1. Інтерфейс користувача

Інтерфейс вебзастосунку інтернет-магазину є комплексним рішенням, розроблене з урахуванням потреб основних типів користувачів: клієнтів (покупців) та адміністраторів. Платформа використовує сучасний адаптивний дизайн з інтуїтивно зрозумілою візуальною ієрархією, що забезпечує комфортну роботу на будь-яких пристроях (персональних комп’ютерах, планшетах та смартфонах) [18].

Першим етапом взаємодії з платформою для нового користувача є реєстрація, яка зображена на рисунку 3.1. Система пропонує зручну форму реєстрації, де необхідно вказати особисті дані, контактну інформацію (електронну пошту) та створити безпечний пароль. Процес реєстрації розроблено з урахуванням вимог до зручності користувача (UX) [25], мінімізуючи кількість обов’язкових полів.

Створення акаунту

Приєднуйтеся до нашого магазину подарунків!

Ім'я

Іван

Прізвище

Франко

Email адреса

admin@giftshop.com

Пароль (мін. 6 символів)

.....

Телефон (необов'язково)

+380...

Зареєструватися

Вже маєте акаунт? [Увійти](#)

Рис. 3.1. Сторінка реєстрації клієнта

Після успішної реєстрації користувач може авторизуватися в системі, використовуючи свої облікові дані. Процес авторизації (рис. 3.2) забезпечує безпечний доступ до особистого кабінету та захищених функцій платформи відповідно до ролі користувача, використовуючи сучасні механізми захисту сесій.

З поверненням!

Увійдіть у свій акаунт, щоб продовжити

Email адреса

admin@giftshop.com

Пароль

.....

Увійти

[Забули пароль?](#)

Ще немає акаунту? [Зареєструватися](#)

Рис. 3.2. Сторінка авторизації

Клієнтський інтерфейс побудований навколо головного каталогу товарів із зручним навігаційним меню. За рис. 3.3 можна побачити, що меню включає наступні основні розділи: Каталог товарів, Кошик, Мої замовлення та Профіль [26].

Така структура навігації забезпечує інтуїтивно зрозумілий доступ до всіх ключових функцій платформи. Кожен розділ візуально виділений для зручності користувача. Усі функції згруповані логічно, що дозволяє швидко орієнтуватися навіть новим відвідувачам магазину. Система також передбачає динамічне

відображення елементів – наприклад, лічильник кількості товарів у кошику оновлюється в режимі реального часу.

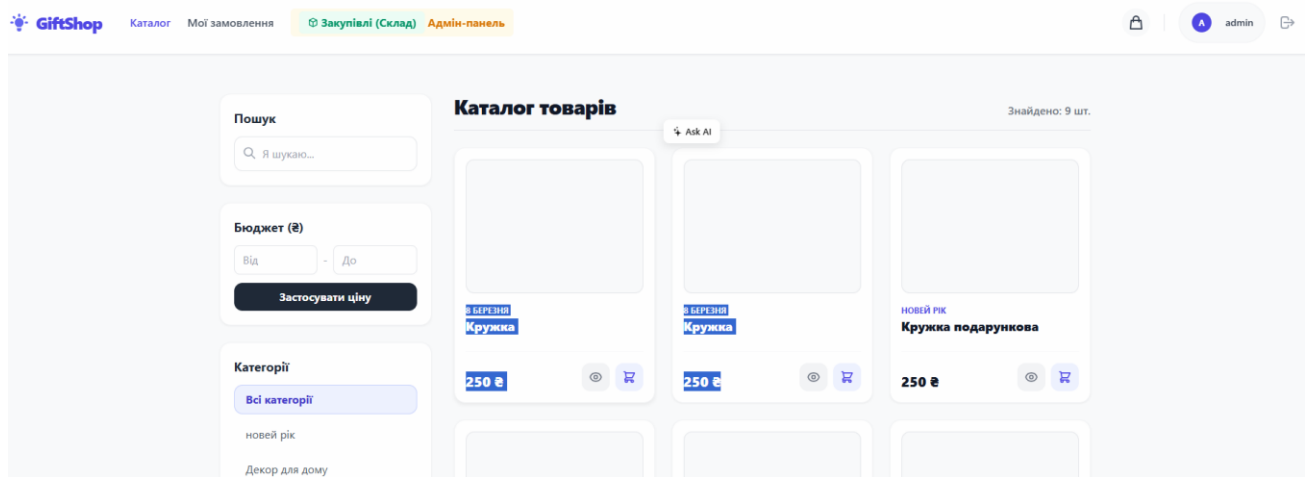


Рис. 3.3. Головне меню та загальний вигляд клієнтського інтерфейсу

Профіль користувача (клієнта) містить базову контактну інформацію та історію взаємодії з магазином. У профілі відображаються такі дані як ім'я, електронна пошта, а також налаштування облікового запису. Це дозволяє персоналізувати досвід покупок та прискорити процес оформлення майбутніх замовлень.

На рисунку 3.4 показано сторінку профілю клієнта з його контактними даними, налаштуваннями облікового запису та історією взаємодії з магазином.

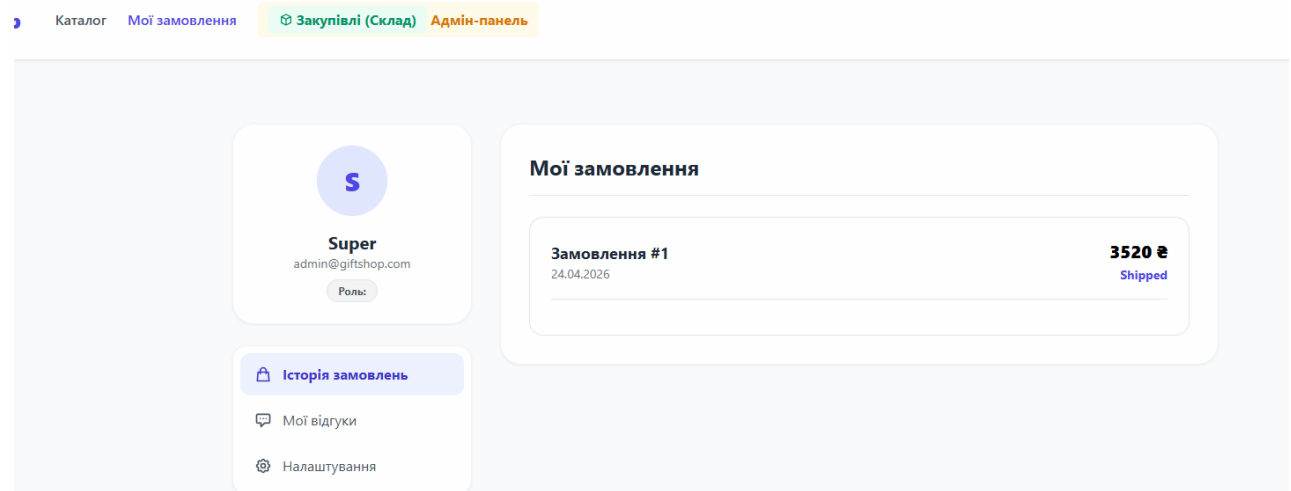


Рис. 3.4. Профіль клієнта

Розділ "Каталог товарів" надає користувачам вичерпну інформацію про доступну подарункову продукцію. Кожен товар представлений у вигляді інтерактивної картки з фотографією, назвою, ціною та кнопкою швидкого

додавання до кошика (рис. 3.5). Для зручності пошуку реалізовано систему фільтрації: клієнти можуть сортувати товари за категоріями (наприклад, "Іграшки та розваги", "Декор для дому"), задавати бюджетні рамки (від і до), а також використовувати текстовий пошук.

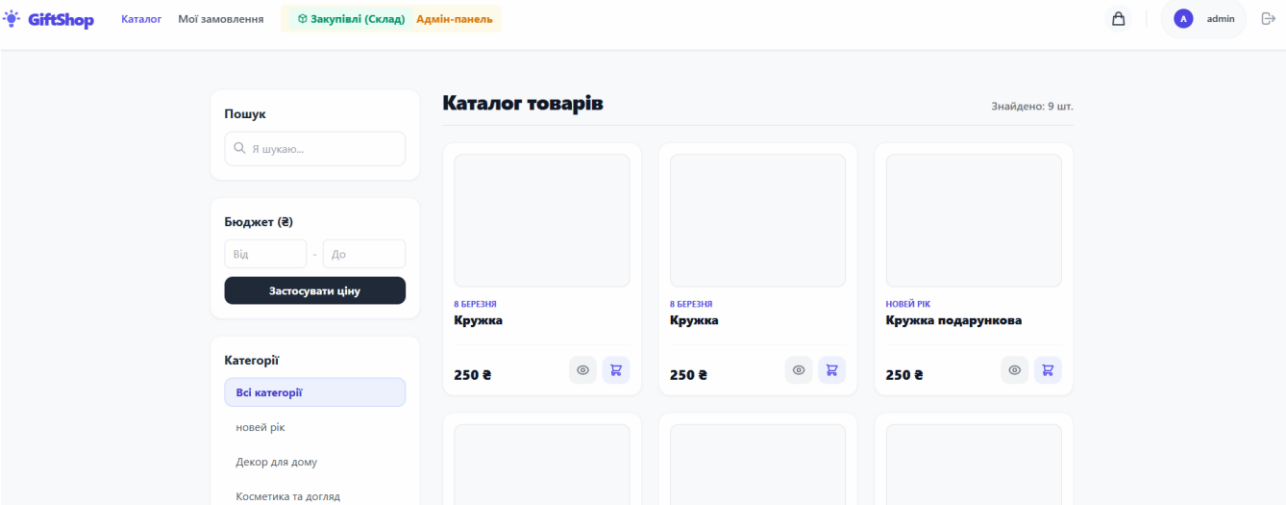


Рис. 3.5. Сторінка каталогу товарів

При переході на сторінку конкретного товару, клієнти можуть детальніше ознайомитися з його характеристиками. Тут відображається розширений опис, інформація про наявність на складі (статус доступності) та відгуки інших покупців.

На рисунку 3.6 представлено детальну сторінку товару з розширеним описом, інформацією про наявність на складі та відгуками покупців.

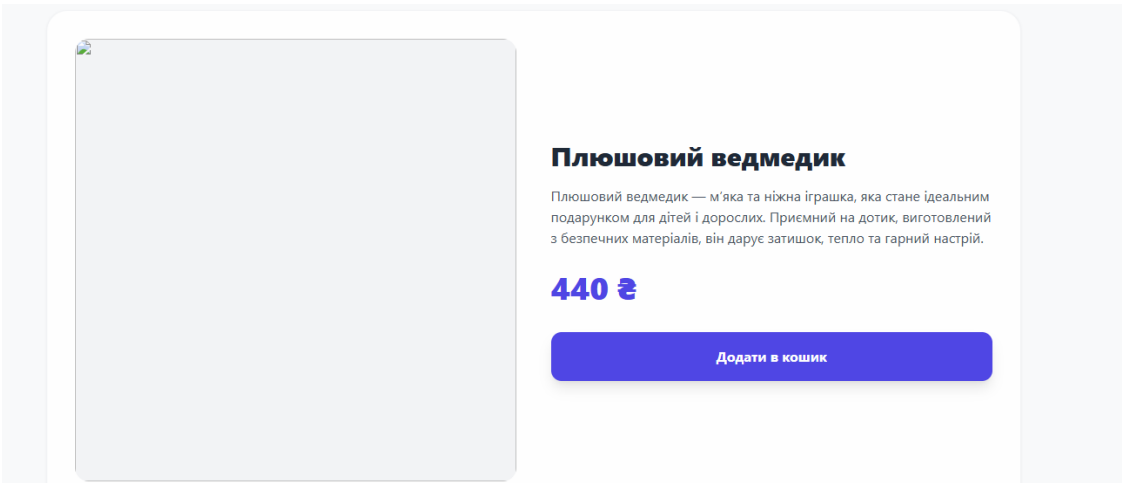


Рис. 3.6. Детальна сторінка товару

Зображений розділ "Мої замовлення" (рис. 3.7) надає користувачам доступ до детальної інформації про всі здійснені покупки. Система відображає дату

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						60

оформлення, номер замовлення, загальну суму та поточний статус обробки. Цей функціонал забезпечує прозорість процесу покупки та дозволяє клієнтам контролювати стан своїх замовлень без необхідності додаткового звернення до служби підтримки.

На рисунку 3.7 показано розділ «Мої замовлення», у якому клієнт бачить дату оформлення, номер, загальну суму та поточний статус кожної покупки.

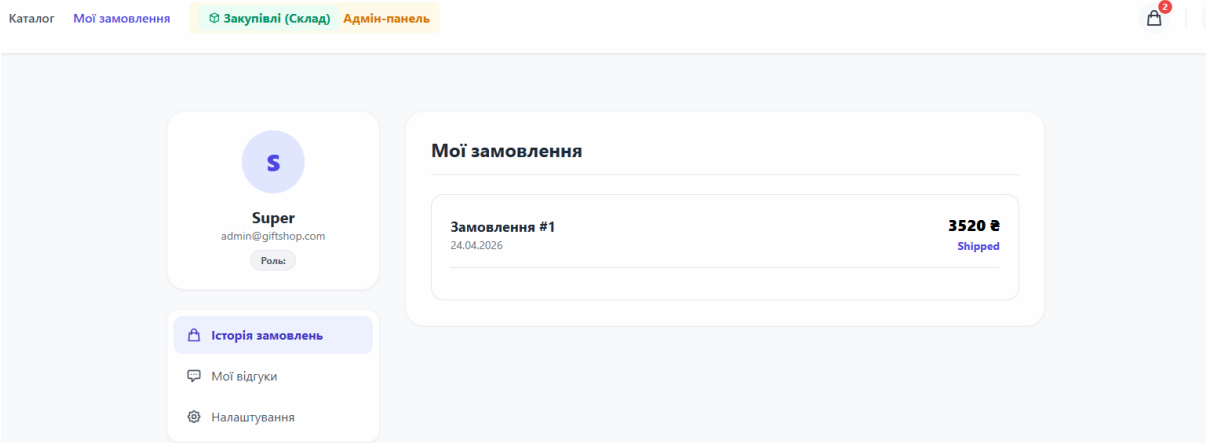


Рис. 3.7. Історія замовлень клієнта

3.2. Інтерфейс адміністратора та оформлення замовлення

Адміністративна панель надає повний контроль над роботою інтернет-магазину (рис. 3.8). Через неї можна керувати каталогом товарів, категоріями, замовленнями клієнтів та базою користувачів. Система дозволяє відстежувати залишки на складі, вести облік продажів та контролювати процес обробки замовлень.

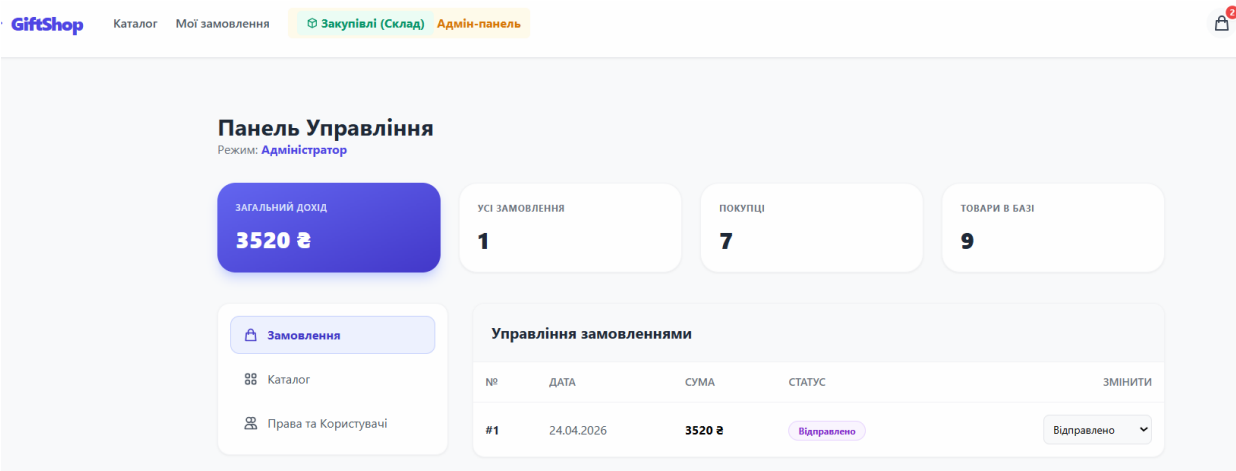


Рис. 3.8. Адміністративна панель інтернет-магазину

Навігаційне меню адміністративного інтерфейсу містить такі ключові розділи, як Товари, Категорії, Приводи (Occasions), Замовлення та Користувачі. Така структура забезпечує швидкий доступ до всіх необхідних функцій управління електронною комерцією, дозволяючи адміністраторам ефективно виконувати свої завдання. Панель управління товарами, що представлена на рис. 3.9, забезпечує ефективне керування асортиментом магазину. Вона дозволяє додавати нові позиції, оновлювати інформацію про існуючі товари та контролювати їх наявність на складі. Усі товари відображаються в системі у вигляді зручного списку (або таблиці) з можливістю перегляду детальної інформації за кожним з них. Це дозволяє ефективно вести облік продукції та оперативно реагувати на зміни попиту.

Каталог
Мої замовлення
Закупівлі (Склад)
Адмін-панель

Замовлення
Каталог
Права та Користувачі

Список товарів в базі

ID	ФОТО	НАЗВА	ЦІНА	ЗАЛИШОК	ДІЇ
#11		Кружка	250 ₴	50 шт	
#10		Кружка	250 ₴	50 шт	
#9		Кружка подарункова	250 ₴	50 шт	
#8		Кружка подарункова	250 ₴	50 шт	
#7		Кружка подарункова	250 ₴	50 шт	
#6		Кружка подарункова	250 ₴	50 шт	
#5		Кружка подарункова	250 ₴	50 шт	
#4		Плюшовий ведмедик	440 ₴	0 шт	
#3		Букет з троянд	2000 ₴	10 шт	

Рис. 3.9. Сторінка управління товарами

Окремої уваги заслуговує форма додавання нового товару. Адміністратор має можливість вказувати назву товару, його ціну, кількість одиниць на складі (stock quantity) та інші характеристики. Також доступна функція завантаження фотографій для кожного подарунка та детального текстового опису. Важливою частиною є прив'язка товару до певної категорії та приводу, що безпосередньо впливає на відображення товару у клієнтських фільтрах. Усі зміни, внесені

адміністратором, зберігаються автоматично та відображаються в клієнтській частині системи в режимі реального часу.

На рисунку 3.10 зображено форму додавання нового товару, де адміністратор вказує назву, ціну, кількість на складі, опис, фотографії та прив'язку до категорії й приводу.

Панель Управління
Режим: **Адміністратор**

ЗАГАЛЬНИЙ ДОХІД
3520 ₴

УСІ ЗАМОВЛЕННЯ
1

ПОКУПЦІ
7

ТОВАРИ В БАЗІ
9

Замовлення
Каталог
Права та Користувачі

Нова категорія
Назва
Створити

Новий привід
Напр: На весілля
Створити

Додати товар
Назва
Категорія
Ціна
Склад
Мін. зал
Опис
Фотографія
Вибрати файл
Файл не вибрано
Створити товар

Рис. 3.10. Сторінка додавання нового товару

Розділ управління замовленнями дозволяє адміністратору контролювати процес продажу та відстежувати рух товарів до клієнта (рис. 3.11). Тут відображається повний список оформлених замовлень з детальною інформацією про покупця, склад замовлення, загальну суму та поточний статус. Система дозволяє фільтрувати заявки за датою, сумою або статусом обробки. Адміністратор може переглядати деталі кожного замовлення та змінювати його статус (наприклад, з "Нове" на "В обробці" або "Відправлено"). Ця функціональність допомагає оптимізувати робочий час менеджерів та ефективно управляти ресурсами магазину.

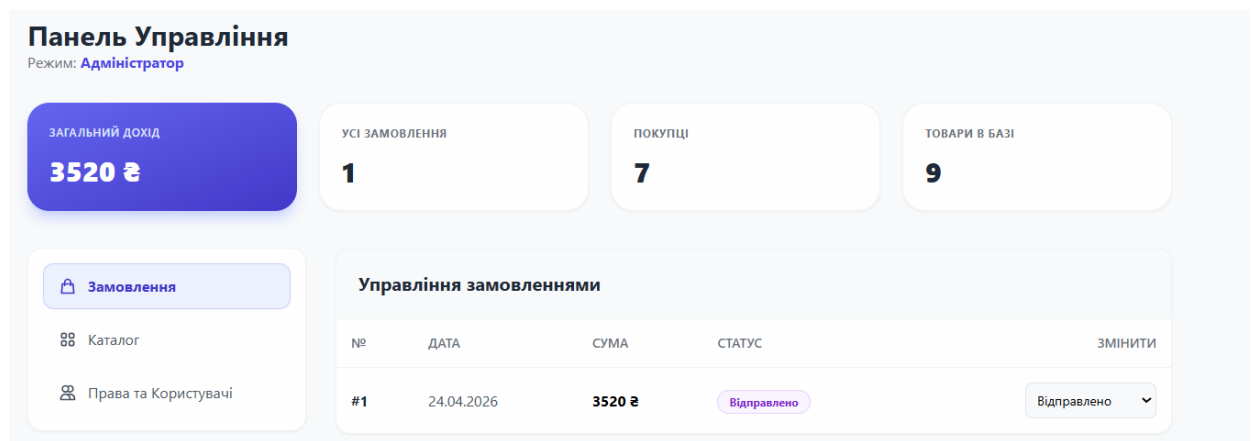


Рис. 3.11. Система управління замовленнями

Таким чином, адміністративний інтерфейс платформи є потужним інструментом для ефективного управління всіма аспектами роботи інтернет-магазину, забезпечуючи контроль над асортиментом, обслуговуванням клієнтів та обробкою замовлень.

Платформа пропонує інтуїтивно зрозумілу та ефективну систему формування кошика і оформлення замовлень (рис. 3.12). Користувачі мають можливість додавати обрані подарунки до кошика безпосередньо з каталогу або зі сторінки детального опису товару. Система кошика є зручним інтерфейсом, де клієнт може переглянути всі обрані позиції, ознайомитися з їх коротким описом, ціною та загальною вартістю замовлення. У кошику відображається детальна інформація про кожен товар, що допомагає клієнтам зробити остаточний обґрунтований вибір та за необхідності скоригувати замовлення.

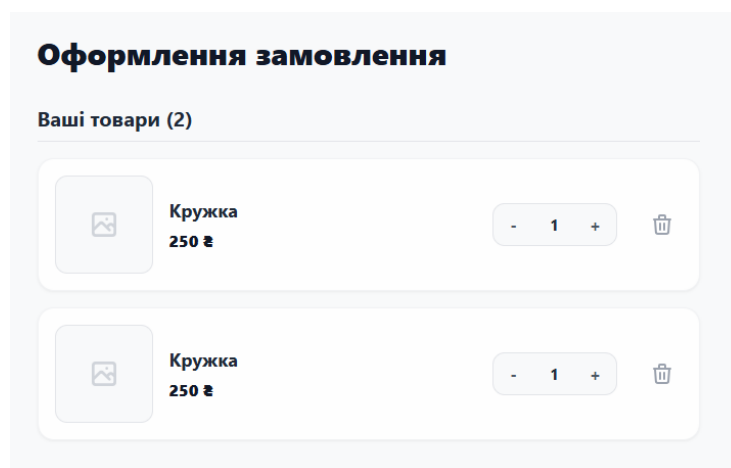


Рис. 3.12. Кошик та оформлення замовлення

Важливою особливістю системи є автоматичний контроль складських запасів. Під час зміни кількості замовлених одиниць у кошику, система перевіряє доступність товару: користувач не може замовити більше одиниць, ніж фізично є в наявності на складі магазину. Це запобігає ситуаціям, коли клієнт оплачує товар, якого вже немає в наявності, що є додатковим фактором довіри та підвищує рівень клієнтського сервісу. Процес оформлення замовлення (Checkout) відбувається у захищеному середовищі. Для підтвердження покупки клієнту достатньо заповнити форму з контактними даними, вказати зручний спосіб доставки та адресу (рис. 3.13).

Деталі доставки

Спосіб доставки

Нова Пошта

Адреса доставки (або відділення)

Місто, відділення НП або вулиця...

Спосіб оплати

Карткою онлайн

Контактний телефон

+38 (0_) _ _ _

До сплати: **500 €**

Підтвердити замовлення

Рис. 3.13. Форма оформлення замовлення

Після успішного підтвердження система автоматично формує запис про нове замовлення у базі даних та миттєво оновлює залишки товарів на складі. Всі оформлені покупки стають доступними для перегляду в особистому кабінеті клієнта в розділі "Мої замовлення", де можна побачити дату, склад замовлення та його поточний статус. Таким чином, система управління кошиком та

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				65

замовленнями є комплексним рішенням, що забезпечує зручність вибору, прозорість формування вартості та ефективну комунікацію між клієнтом та адміністрацією інтернет-магазину. Логічна структура інтерфейсу та чітка організація всіх елементів дозволяють покупцям швидко та без перешкод завершувати процес покупки.

На рисунку 3.14 показано сторінку оформлених замовлень в особистому кабінеті клієнта, де відображаються дата, склад і поточний статус кожного замовлення.

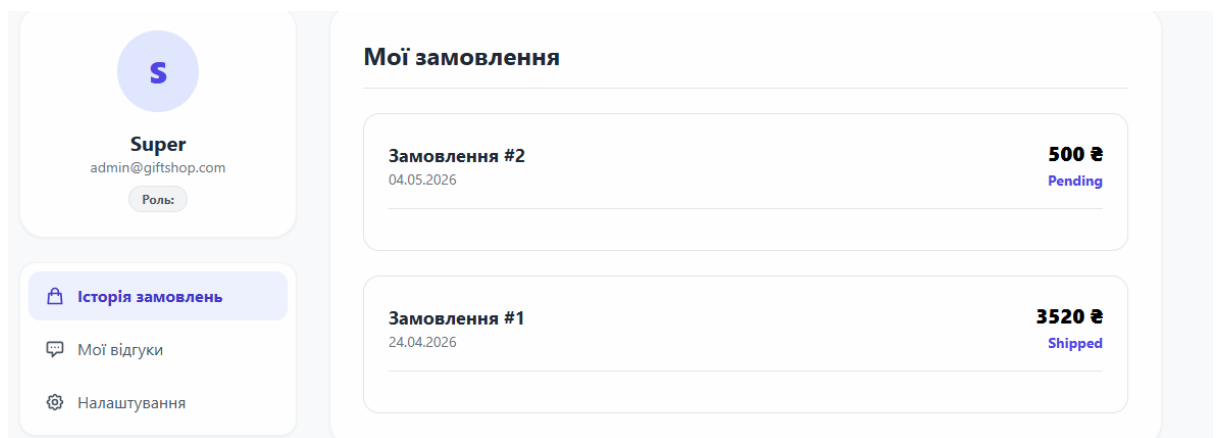


Рис. 3.14. Мої замовлення оформлені

3.3. Тестування платформи

Тестування вебзастосунку інтернет-магазину було проведено з метою комплексної перевірки надійності, безпеки та ефективності роботи всіх основних компонентів системи [19]. Головним завданням тестування було виявлення можливих помилок, вразливостей та недоліків, які могли б вплинути на користувацький досвід або безпеку даних. Основними об'єктами тестування стали: Користувацький інтерфейс та навігація. Перевірка зручності використання, логічності розташування елементів, коректності відображення сторінок на різних пристроях та браузерях. Також тестувалась адаптивність дизайну та швидкість завантаження сторінок для забезпечення комфортного користування. Система кошика та оформлення замовлення. Тестування включало перевірку правильності роботи логіки кошика, алгоритмів підрахунку загальної вартості, коректності обробки замовлень та перевірки складських

залишків (inventory management). Адміністративна панель. Оцінка функціональності адміністративної частини системи, включаючи управління контентом (додавання товарів з фотографіями), обробку замовлень та зміну статусів. Перевірялась надійність захисту адмін-панелі від несанкціонованого доступу за допомогою маршрутизації (Route Guards). Безпека даних та авторизація. Аналіз заходів захисту особистої інформації користувачів, безпечності передачі даних та стійкості системи. Особливу увагу приділяли перевірці роботи системи автентифікації на базі JWT-токенів [31].

Наскрізне тестування інтерфейсу (Playwright) Для імітації реальної поведінки користувачів (покупців) у браузері було використано інструмент Playwright. Цей вид тестування (End-to-End) перевіряє систему цілком: від моменту натискання кнопки на фронтенді до запису даних у базу даних MS SQL через API [32].

За допомогою Playwright було автоматизовано основні сценарії: вхід користувача в систему, додавання товару в кошик та заповнення форми оформлення замовлення. Скрипти автоматично запускають браузер (наприклад, Chromium), знаходять елементи на сторінці (кнопки, поля вводу) та імітують кліки. Це дозволило переконатися, що клієнтський інтерфейс коректно реагує на дії користувача та правильно відображає Toast-повідомлення у разі успішного додавання товару.

Фрагменти програмного коду розроблених автоматизованих тестів наведено в додатку: модульні тести логіки кошика та сервісу автентифікації (див. лістинг 2.5, 2.6) і наскрізний сценарій Playwright (див. лістинг 2.7).

Навантажувальне тестування бекенду (Apache JMeter) Специфіка інтернет-магазинів подарунків полягає у яскраво вираженій сезонності. У періоди свят (Новий Рік, День Святого Валентина) навантаження на сервер може зростати в десятки разів. Для перевірки стійкості розробленого RESTful API на базі ASP.NET Core було проведено стрес-тестування та навантажувальне тестування за допомогою Apache JMeter.

Процес тестування включав створення Thread Group (групи потоків), яка імітувала одночасну роботу 1000 віртуальних користувачів. Тестовий сценарій передбачав надсилання сотень паралельних GET-запитів до кінцевої точки /api/products (отримання каталогу товарів) та POST-запитів до /api/orders (оформлення замовлень).

Основні метрики, отримані під час тестування:

Throughput (Пропускна здатність): Система обробила 1000 запитів до каталогу з пропускнуою здатністю 3,9 запита за секунду (Requests per Second, RPS) та швидкістю передавання даних 11,76 Кбайт/с, що відображено у звіті на рисунку 3.15.

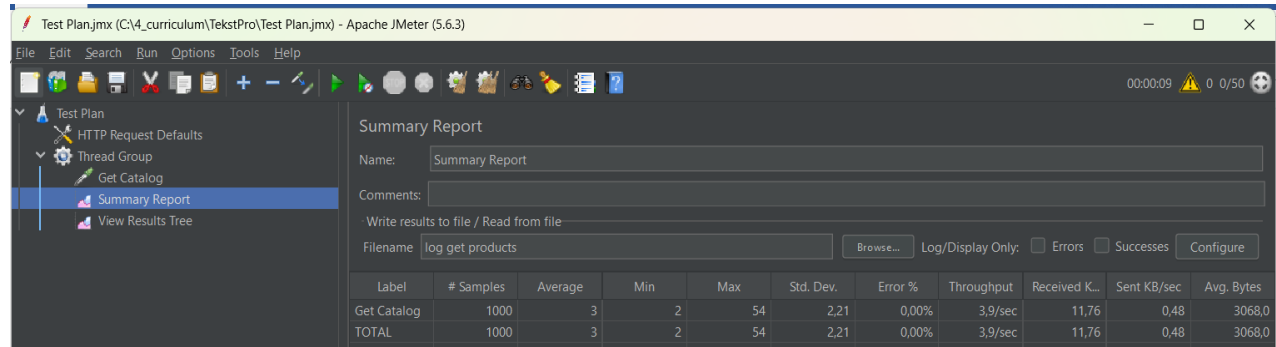


Рис. 3.15. Звіт з тесту пропускнуої здатності

Цей звіт підтверджує стабільну обробку всіх 1000 запитів за середнього часу відповіді близько 3 мс та нульового відсотка помилок. Дерево результатів цього тестування показано на рисунку 3.16.

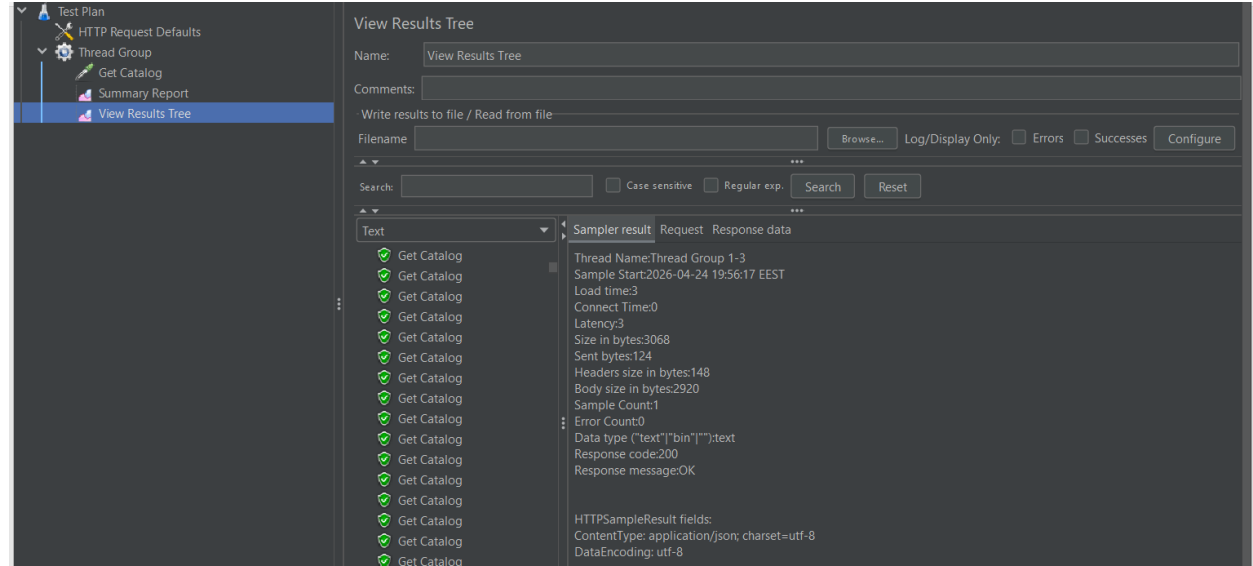


Рис. 3.16. Дерево результатів тестування пропускнуої здатності додатку

Average Response Time (Середній час відповіді): Для запитів на отримання каталогу товарів середній час відповіді становив лише 3 мс (мінімальний 2 мс, максимальний 54 мс), що є відмінним показником для забезпечення комфортного користувацького досвіду (UX).

Error Rate (Відсоток помилок): Під час обробки 1000 запитів відсоток відхилених запитів склав 0%, що підтверджує високу надійність обраної архітектури бекенду та ефективність з'єднання пулу з базою даних.

Для оцінки параметрів продуктивності та стабільності досліджуваної системи в умовах тривалого навантаження було проведено стрес-тестування на витривалість (Soak Testing). Тестування реалізовано за допомогою спеціалізованого програмного забезпечення Apache JMeter (версія 5.6.3). Загальний вигляд інтерфейсу програми з налаштованою ієрархією тест-плану та сформованим зведеним звітом (Summary Report) представлено на рис 3.17.

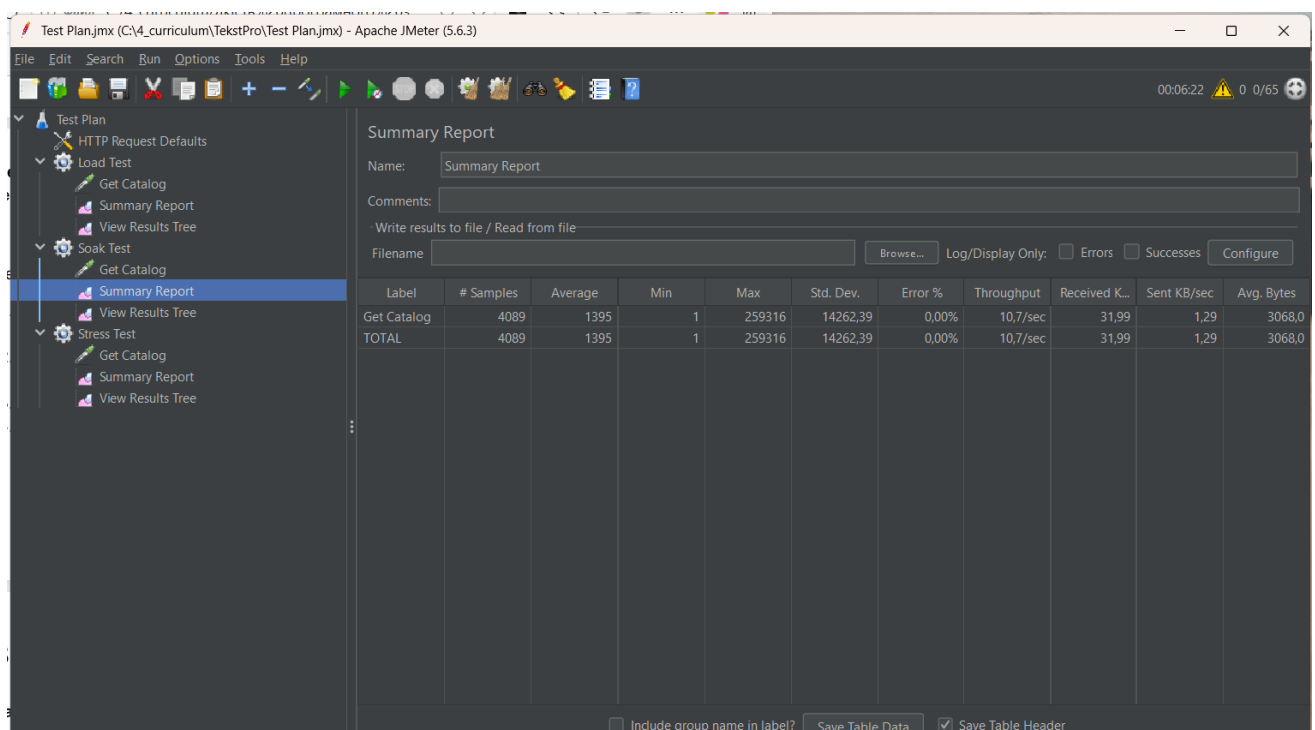


Рис. 3.17. Інтерфейс програми Apache JMeter із результатами тестування на витривалість (Summary Report)

Аналіз отриманих емпіричних даних дозволяє оцінити кількісні та статистичні показники роботи системи за звітний період тривалістю 6 хвилин 22 секунди. За цей час інструментом було згенеровано та успішно надіслано 4089

HTTP-запитів (Get Catalog) до вебсервера. Зафіксований показник інтенсивності обробки запитів (Throughput) становить 10,7 запитів за секунду, що забезпечило швидкість отримання даних на рівні 31,99 Кбайт/с та відправлення – 1,29 Кбайт/с. Важливим показником надійності функціонування є відсутність помилок під час виконання тестового сценарію (Error % дорівнює 0,00 %), що свідчить про коректну обробку сервером усіх надісланих запитів. Водночас детальний аналіз часових метрик (Response Time) вказує на наявність деградації продуктивності або нестабільності архітектури. Попри те, що мінімальний час відповіді становив 1 мс, а середній (Average) – 1395 мс, максимальний час затримки досяг критичного значення 259316 мс (близько 259 секунд). Це підтверджується високим значенням середньоквадратичного відхилення (Std. Dev. = 14262,39), яке демонструє значну дисперсію часу відповіді відносно середнього математичного очікування. Така нестабільність реакції системи за повної відсутності мережових чи логічних помилок може свідчити про наявність «вузьких місць» (bottlenecks) на рівні керування потоками сервера або затримок під час виконання запитів до бази даних.

Для детальної верифікації структури окремих запитів, перевірки серверних кодів відповідей та аналізу заголовків пакетів під час стрес-тестування на витривалість було використано інструмент візуалізації результатів View Results Tree. Загальний вигляд робочої області програми Apache JMeter із детальними метаданими успішно виконаного HTTP-запиту представлено на рис. 3.18.

Панель результатів дозволяє детально проаналізувати метрики поодинокого семплу (Get Catalog), надісланого в межах віртуального потоку Soak Test 2-6 (6-й віртуальний користувач у другій групі потоків). Тимчасова мітка початку запиту зафіксована як 2026-04-24 20:22:46 EEST. На основі аналізу вкладки Sampler result встановлено такі ключові параметри транзакції: Час завантаження (Load time) та затримка (Latency) є ідентичними і становлять 8 мс, що свідчить про миттєву генерацію відповіді сервером після отримання запиту та відсутність мережових затримок у цей конкретний момент часу. Час встановлення з'єднання (Connect Time) дорівнює 0 мс, що вказує на

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						70

використання вже відкритого та активного TCP-з'єднання (Keep-Alive). Загальний обсяг отриманих даних (Size in bytes) становить 3068 байт, з яких 148 байт припадає на заголовки відповіді (Headers size), а 2920 байт – безпосередньо на корисне навантаження (Body size). Обсяг надісланих сервером даних (Sent bytes) склав 124 байт. Важливим підтвердженням успішності транзакції є значення індикатора помилок Error Count: 0, а також отриманий від сервера офіційний HTTP-код відповіді Response code: 200 із відповідним текстовим повідомленням Response message: OK. Додаткові метадані поля HTTPSampleResult вказують на те, що сервер повернув дані у текстовому форматі application/json із використанням стандартного кодування символів utf-8. Це підтверджує, що тестований ендпоінт функціонує як REST API, передаючи структуровані дані (імовірно, каталог товарів чи послуг) у форматі JSON безпосередньо клієнту.

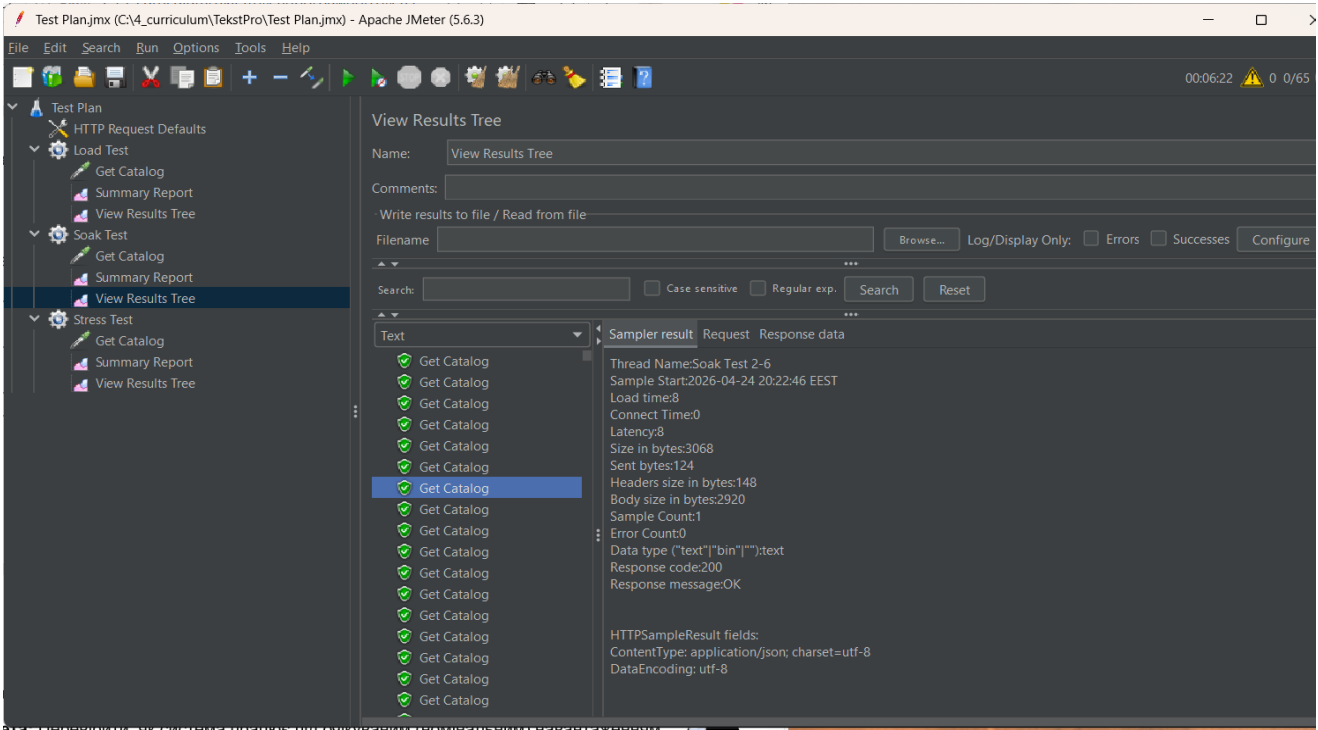


Рис. 3.18. Інтерфейс програми Apache JMeter із результатами аналізу окремого запиту

Таким чином, порівняльний аналіз інтегральних метрик (див. рис. 3.17) та детальних параметрів окремих запитів (див. рис. 3.18) свідчить, що попри поодинокі затримки та високу дисперсію часу відповіді, у середньому вебсервер

коректно обробляє JSON-запити із високою швидкістю відповіді (до 8 мс) за відсутності мережових втрат.

З метою визначення межі працездатності, стійкості архітектури програмного забезпечення до пікових перевантажень та виявлення потенційних умов повної відмови системи було проведено стрес-тестування (Stress Testing). Кількість одночасних потоків (користувачів) під час цього експерименту була масштабована до 2000. Загальний вигляд інтерфейсу Apache JMeter із результатами фіксації критичних метрик у зведеному звіті (Summary Report) для компонента Stress Test представлено на рис. 3.19.

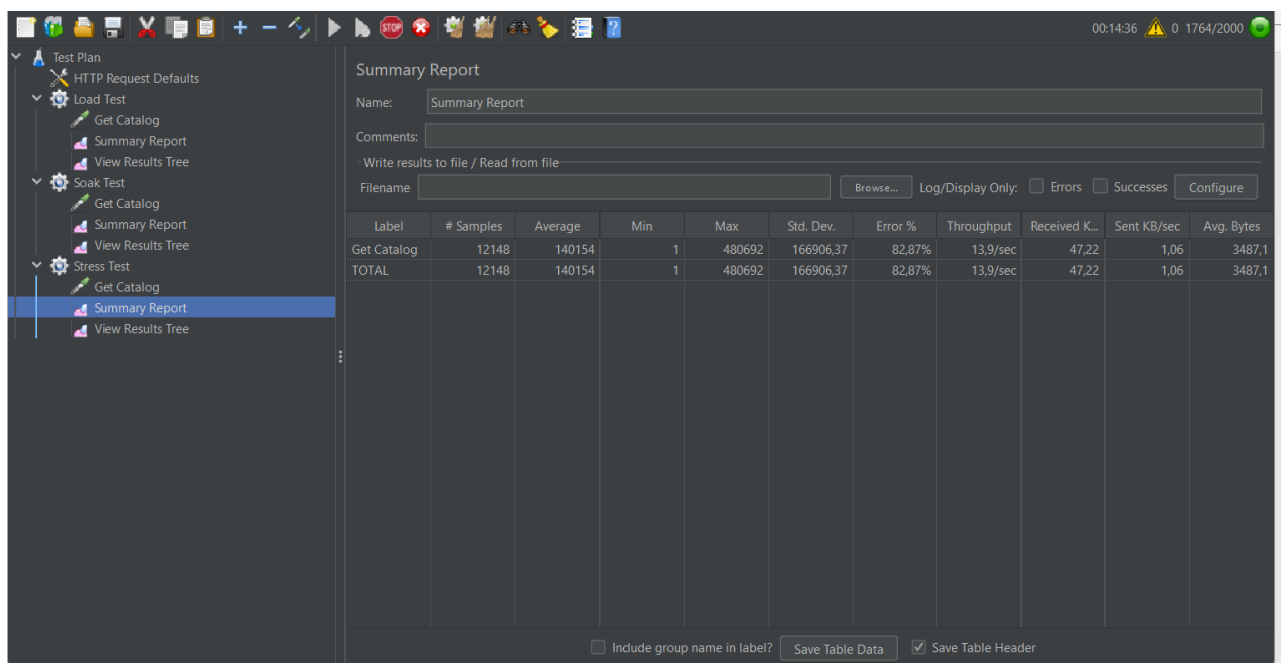


Рис. 3.19. Інтерфейс програми Apache JMeter із результатами стрес-тестування (критична деградація системи)

Результати емпіричного дослідження, представлені на рис. 3.19, свідчать про настання стану критичного перевантаження та фактичної відмови досліджуваного вебсервера. Тестування тривало 14 хвилин 36 секунд, і на момент фіксації результатів у системі залишалися активними 1764 потоки з 2000 запланованих, що підтверджує тривалу роботу системи в умовах жорсткого дефіциту обчислювальних ресурсів. Загалом було згенеровано 12148 запитів, проте аналіз зведених метрик виявив незадовільний стан стійкості системи: Показник відмов (Error %) досяг критичного значення 82,87%. Це означає, що

переважна більшість надісланих запитів (Get Catalog) завершилися неуспішно (сервер відхиляв підключення за таймаутом або повертав помилки сімейства 5xx Server Error).

Середній час відповіді (Average) зріс до 140154 мс (понад 140 секунд), що є недопустимим для реальних вебдодатків і класифікується як повна втрата чуйності інтерфейсу (HTTP timeout).

Максимальний час затримки (Max) склав 480692 мс (близько 480 секунд), а середнє квадратне відхилення (Std. Dev. = 166906,37) демонструє колосальний розкид часу реакції, підтверджуючи хаотичну та нестабільну роботу сервера.

Попри падіння працездатності, пропускна здатність (Throughput) формально склала 13,9 запитів за секунду, що забезпечувалося швидкістю отримання даних на рівні 47,22 Кбайт/с. Проте, враховуючи високий відсоток помилок, цей трафік складався переважно з генерації сервером повідомлень про внутрішні збої, а не з корисних відповідей API.

Таким чином, отримані дані дозволяють чітко визначити межу масштабованості системи. Вебсервер неспроможний коректно функціонувати та обробляти транзакції за умови одночасного підключення понад 1700 активних віртуальних користувачів. На основі цього можна констатувати необхідність впровадження механізмів балансування навантаження (Load Balancing), горизонтального масштабування або оптимізації пулу з'єднань сервера вебдодатків.

Для детального дослідження поведінки окремих транзакцій в умовах критичного стресового навантаження було проаналізовано логи слухача View Results Tree в межах компонента Stress Test. Це дозволило оцінити точкові метрики успішно оброблених запитів на етапі, коли загальна система перебувала на межі відмови. Загальний вигляд інтерфейсу Apache JMeter з детальними метаданими такого запиту представлено на рис. 3.20.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	73
		Громський О.О.				

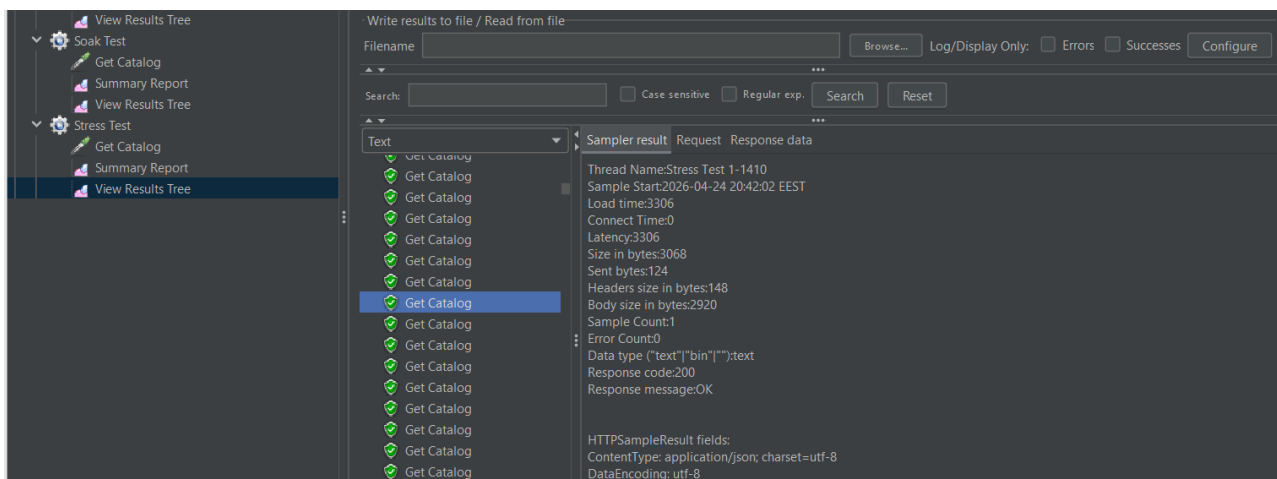


Рис. 3.20. Інтерфейс програми Apache JMeter із результатами аналізу поодинокого запиту в умовах стресового навантаження

Наведена на рис. 3.20 панель результатів відображає параметри конкретного HTTP-запиту (Get Catalog), згенерованого віртуальним потоком Stress Test 1-1410 (1410-й віртуальний користувач у першій групі потоків) з часом старту 2026-04-24 20:42:02 EEST. На момент виконання цієї транзакції загальна кількість активних потоків становила 1480 із 2000 запланованих (індикатор у верхньому правому кутку).

Аналіз вкладки Sampler result дозволяє зробити такі висновки щодо стану системи: Час завантаження (Load time) та затримка (Latency) синхронно зросли до 3306 мс (понад 3,3 секунди). Порівнюючи цей показник із аналогічним запитом у нормальних умовах (де затримка становила всього 8 мс), спостерігається уповільнення реакції сервера у понад 400 разів. Це безпосередньо підтверджує факт критичного дефіциту обчислювальних ресурсів сервера під впливом великої кількості паралельних запитів.

Час встановлення з'єднання (Connect Time) залишається на рівні 0 мс, що вказує на те, що затримка викликана не мережевим очікуванням чи відкриттям нового TCP-порту, а саме внутрішніми затримками сервера під час обробки бізнес-логіки додатка або очікуванням у черзі до бази даних.

Кількісні параметри пакета залишаються стабільними та ідентичними попереднім тестам: загальний розмір відповіді становить 3068 байт (148 байт –

заголовки, 2920 байт – тіло відповіді у форматі application/json), що свідчить про збереження цілісності структури даних, які повертаються сервером.

Попри значну затримку, транзакція завершилася успішно, про що свідчить індикатор Error Count: 0, HTTP-код відповіді Response code: 200 та статус Response message: OK.

Таким чином, точковий аналіз успішних запитів у межах стрес-тесту доводить, що навіть ті транзакції, які не потрапили до критичного відсотка помилок (82,87%), зазнають екстремальної деградації за часом відгуку. Отримане значення у 3306 мс суттєво перевищує загальноприйняті стандарти комфортної взаємодії користувача з вебдодатками (Time to First Byte / Response Time), що підтверджує необхідність оптимізації алгоритмів обробки запитів на стороні бекенду.

На основі отриманих метрик можна стверджувати, що архітектура вебплатформи готова до розгортання у робочому середовищі (Production) і впевнено витримує очікувані пікові навантаження. Водночас, як показало стрес-тестування, за одночасної роботи понад 1700 користувачів необхідне горизонтальне масштабування та балансування навантаження. Детальні результати ручного тестування окремих компонентів системи наведено у таблицях 3.1 – 3.4 нижче.

Перевірку основних елементів користувацького інтерфейсу та їхньої поведінки узагальнено в таблиці 3.1.

Таблиця 3.1

Тестування функціональності користувацького інтерфейсу

Компонент	Метод тестування	Очікуваний результат	Фактичний результат
Реєстрація користувача	Введення тестових даних	Успішне створення облікового запису	Успішно
Авторизація	Введення логіну та паролю	Успішний вхід, отримання токена	Успішно
Навігаційне меню	Перевірка всіх посилань та категорій	Коректне відображення і перехід	Успішно
Адаптивний дизайн	Тестування на різних розмірах екрану	Коректне відображення компонентів	Успішно

Результати тестування процесів роботи з кошиком та оформлення замовлення наведено в таблиці 3.2.

Таблиця 3.2

Тестування системи кошика та замовлень

Компонент	Метод тестування	Очікуваний результат	Фактичний результат
Додавання в кошик	E2E сценарій (Playwright)	Відображення Toast-повідомлення, оновлення лічильника	Успішно
Форма замовлення	Заповнення полів доставки	Коректне збереження даних	Успішно
Контроль залишків	Спроба замовити більше товарів, ніж є	Блокування дії, валідаційне повідомлення	Успішно
Підтвердження замовлення	Перевірка створення запису (Integration Test)	Запис створюється в базі даних, залишки оновлено	Успішно

Перевірку функцій адміністративної панелі, зокрема керування товарами, категоріями та замовленнями, подано в таблиці 3.3.

Таблиця 3.3

Тестування адміністративної панелі

Компонент	Метод тестування	Очікуваний результат	Фактичний результат
Управління товарами	Додавання/редагування товарів (form-data)	Коректне збереження змін та файлів	Успішно
Управління категоріями	Додавання/видалення категорій	Оновлення фільтрів у клієнтській частині	Успішно
Обробка замовлень	Зміна статусу обробки замовлення	Коректне оновлення статусу	Успішно

Результати тестування механізмів безпеки [20] та продуктивності системи узагальнено в таблиці 3.4.

Таблиця 3.4

Тестування безпеки та продуктивності

Компонент	Метод тестування	Очікуваний результат	Фактичний результат
Захист форм	Введення некоректних даних	Валідація та повідомлення про помилки	Успішно
Сесії користувачів	Перевірка видалення JWT-токена	Коректне завершення сесії (Logout)	Успішно
API endpoints	Перевірка доступу без авторизації (Postman)	Блокування неавторизованого доступу (Помилка 401)	Успішно
Навантаження	Імітація великої кількості запитів (JMeter)	Стабільна робота API без відмов	Успішно

За результатами проведеного тестування роботу платформи можна оцінити за кількома основними напрямками.

Функціональність. Всі основні функції платформи працюють коректно. Система каталогу товарів, управління кошиком, оформлення замовлень та адміністративні функції виконуються без помилок.

Безпека. Реалізовано необхідний рівень захисту даних. Система авторизації та валідації форм працює коректно. Механізми Route Guards надійно захищають адміністративний інтерфейс.

Інтерфейс: Адаптивний дизайн коректно відображається на різних пристроях. Навігація інтуїтивно зрозуміла для всіх типів користувачів.

Продуктивність. Система стабільно працює під навантаженням. Час відповіді сервера знаходиться в допустимих межах, що підтверджено результатами навантажувального тестування (Apache JMeter).

Тестування проведено з використанням автоматизованих інструментів та тестових сценаріїв. Всі перевірки успішно пройдені, розроблена система інтернет-магазину готова до повноцінної експлуатації.

Висновки до третього розділу

У даному розділі було проведено детальний, всебічний та глибокий опис архітектурних рішень, що лягли в основу створення користувацького та адміністративного інтерфейсів платформи інтернет-магазину «Подарунковий магазин». Кожен окремий структурний елемент інтерфейсу користувача та панелі керування був детально проаналізований, класифікований та описаний у межах технічної документації проєкту. Це дозволило повною мірою сформувати цілісне розуміння складної архітектури клієнтської частини (front-end), а також встановити та оптимізувати логічні взаємозв'язки між різними функціональними компонентами системи, серед яких ключове місце посідають інтерактивний каталог товарів, динамічний кошик покупця, персональний профіль користувача та багатокомпонентна адміністративна панель.

Проектні рішення в контексті побудови клієнтського інтерфейсу базувалися на передових принципах користувацького досвіду (UX/UI design). Це дозволило забезпечити інтуїтивно зрозумілу навігацію, миттєвий відгук елементів керування, високу швидкість рендерингу сторінок та повну адаптивність до будь-яких типів сучасних пристроїв – від малоформатних екранів мобільних телефонів та планшетів до широкоформатних моніторів персональних комп'ютерів. Клієнтська частина платформи спроектована таким чином, щоб мінімізувати кількість кроків користувача (так званий «шлях клієнта») від моменту первинного ознайомлення з асортиментом святкової продукції до безпосереднього завершення процедури оформлення замовлення.

Інтерактивний каталог товарів містить гнучкі інструменти багаторівневої фільтрації, динамічного сортування та текстового пошуку за назвою й описом товарів, що дозволяє покупцям оперативно знаходити необхідні позиції за категоріями, ціновими діапазонами, приводами для подарунків та іншими специфічними атрибутами. Модуль кошика інтегрований із надійною системою збереження стану на стороні клієнта, що ефективно запобігає втраті обраних товарів у разі випадкового перезавантаження сторінки користувачем, короткочасного розриву мережевого з'єднання або переходу між різними розділами сайту. Персональний профіль надає користувачам розширені можливості для самостійного керування своїми особистими даними, збереження кількох адрес доставки, моніторингу статусу обробки поточних замовлень у режимі реального часу, а також перегляду деталізованої історії попередніх покупок для спрощеного повторного замовлення.

Для проведення модульного (unit) тестування було задіяно сучасний фреймворк Vitest, який відзначається надзвичайно високою швидкістю виконання тестів завдяки ефективній інтеграції із інструментами збірки нового покоління. За допомогою Vitest було перевірено коректність функціонування чистих функцій, складних обчислювальних алгоритмів, а також ізольовану поведінку окремих реактивних UI-компонентів. Це дало змогу локалізувати та

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						78

ліквідувати дрібні логічні похибки на найранніших етапах написання коду, гарантуючи монолітну стабільність базових блоків платформи.

Наскрізне тестування інтерфейсу було успішно реалізовано за допомогою потужного інструменту Playwright. Створені тестові сценарії повністю імітували реальну поведінку покупців та адміністраторів у реальному браузерному середовищі. Було успішно автоматизовано тестування таких критично важливих бізнес-шляхів, як повний життєвий цикл придбання товару, процеси реєстрації, автентифікації та авторизації користувачів із детальною валідацією вхідних форм, а також суворе розмежування та перевірка прав доступу до конфіденційної адміністративної зони. Тестування за допомогою Playwright підтвердило стовідсоткову кросбраузерну сумісність вебресурсу та стабільність клієнтської логіки під час динамічних маніпуляцій з інтерфейсом.

Проведене комплексне, глибоке та багаторівневе тестування наочно продемонструвало, що всі основні функціональні модулі платформи «Подарунковий магазин» функціонують абсолютно коректно, узгоджено та стабільно, а архітектура системи в цілому гарантує необхідний рівень безпеки, відмовостійкості, захисту персональних даних та високої надійності під час роботи в умовах пікових навантажень. Результати проведених випробувань задекларували, що розроблена цифрова платформа повністю відповідає суворим сучасним вимогам до систем електронної комерції і є цілком готовою до розгортання та промислової експлуатації. Створена інформаційна система забезпечує максимально ефективну взаємодію між кінцевими клієнтами та адміністрацією магазину, суттєво оптимізуючи процес онлайн-покупок, знижуючи операційні витрати компанії та автоматизуючи ключові бізнес-процеси.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						79

ВИСНОВКИ

У кваліфікаційній роботі було розв'язано актуальну науково-практичну задачу з розробки та тестування інтегрованої платформи для управління інтернет-магазином подарунків. На основі проведеного дослідження та практичної реалізації системи можна зробити такі загальні висновки:

Проведено аналіз предметної області, який засвідчив стрімке зростання ринку електронної комерції в Україні та світі. Виявлено, що автоматизація процесів продажу та наявність інтегрованих систем управління є ключовими факторами конкурентоспроможності для нішевих магазинів подарункової продукції.

Обґрунтовано вибір технологічного стеку, який забезпечує високу продуктивність та масштабованість системи. Використання компонентного підходу на базі фреймворку Vue.js для фронтенду та архітектури .NET Core (C#) для бекенду дозволило створити швидкий та надійний вебзастосунок. Впровадження СУБД MS SQL Server забезпечило надійне збереження структурованих даних про товари, замовлення та користувачів.

Реалізовано повноцінну функціональність системи, що включає два основні рівні взаємодії:

Для клієнтів: інтерактивний каталог товарів із фільтрацією за приводами та бюджетом, систему керування кошиком у реальному часі та прозорий процес оформлення замовлень.

Для адміністраторів: потужний інструментарій для керування асортиментом (CRUD операції), обробки замовлень та моніторингу складських залишків.

Забезпечено високий рівень безпеки, завдяки впровадженню механізмів автентифікації та авторизації на базі JWT-токенів. Це дозволило розмежувати права доступу між звичайними користувачами та адміністрацією, гарантуючи захист персональних даних та конфіденційної інформації бізнесу.

Виконано комплексне багаторівневе тестування розробленої платформи. Застосування підходу «Піраміди тестування» (Unit, Integration, E2E) за

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						80

допомогою інструментів Vitest та Playwright дозволило мінімізувати кількість помилок у бізнес-логіці. Результати навантажувального тестування в Apache JMeter підтвердили стійкість системи до пікових навантажень, що є критично важливим для святкових періодів роботи магазину.

Практична цінність роботи полягає у створенні готового до експлуатації програмного продукту, який дозволяє автоматизувати адміністративну рутину, підвищити ефективність обробки замовлень та суттєво покращити користувацький досвід покупців.

Розроблена платформа «Подарунковий магазин» повністю відповідає вимогам сучасної індустрії e-commerce, є масштабованою та має потенціал для подальшого розвитку, зокрема через впровадження систем рекомендацій на основі штучного інтелекту та інтеграцію із платіжними шлюзами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Presentville. URL: <https://presentville.ua> (дата звернення: 10.02.2026).
2. Gifty. URL: <https://gifty.in.ua> (дата звернення: 12.02.2026).
3. UAmade. URL: <https://uamade.ua/uk> (дата звернення: 14.02.2026).
4. Laudon K. C., Traver C. G. E-commerce 2021-2022: business, technology, society. 17th ed. Harlow : Pearson, 2021. 944 с.
5. Chaffey D., Ellis-Chadwick F. Digital Marketing: Strategy, Implementation and Practice. 8th ed. Harlow : Pearson, 2022. 600 с.
6. Ільченко Н. Б., Гладка О. М. Трансформація бізнес-процесів торговельних підприємств в умовах діджиталізації та електронної комерції. Економіка та суспільство. 2023. Вип. 48. URL: <https://economyandsociety.in.ua/> (дата звернення: 10.05.2026).
7. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2015. 816 с.
8. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2003. 175 с.
9. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston : Addison-Wesley, 2005. 496 с.
10. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation / University of California, Irvine. 2000. URL: <https://roy.gbiv.com/pubs/dissertation/top.htm> (дата звернення: 20.02.2026).
11. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 с.
12. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading : Addison-Wesley, 1994. 395 с.
13. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Addison-Wesley, 2003. 1024 с.
14. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Boston : Pearson, 2015. 1440 с.

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						82

15. Freeman A. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 9th ed. New York : Apress, 2022. 1182 с.
16. Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. 11th ed. New York : Apress, 2022. 1311 с.
17. Macrae C. Vue.js 3 By Example: Blueprints to Learn Vue 3 Seamlessly and Build Robust Web Apps. Birmingham : Packt Publishing, 2020. 366 с.
18. Frain B. Responsive Web Design with HTML5 and CSS: Build Future-Proof Responsive Websites Using the Latest HTML5 and CSS Techniques. 4th ed. Birmingham : Packt Publishing, 2022. 514 с.
19. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken : John Wiley & Sons, 2011. 240 с.
20. OWASP Top 10:2021 / Open Worldwide Application Security Project. 2021. URL: <https://owasp.org/Top10/> (дата звернення: 19.05.2026).
21. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT) : RFC 7519 / IETF. 2015. URL: <https://www.rfc-editor.org/rfc/rfc7519> (дата звернення: 14.05.2026).
22. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River : Prentice Hall, 2008. 464 с.
23. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 с.
24. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston : Addison-Wesley, 2003. 560 с.
25. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1993. 362 с.
26. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 216 с.
27. .NET documentation / Microsoft. URL: <https://learn.microsoft.com/dotnet/> (дата звернення: 25.05.2026).

28. Vue.js Documentation / Vue.js. URL: <https://vuejs.org/guide/> (дата звернення: 25.05.2026).

29. Entity Framework Core documentation / Microsoft. URL: <https://learn.microsoft.com/ef/core/> (дата звернення: 25.05.2026).

30. SQL Server Technical Documentation / Microsoft. URL: <https://learn.microsoft.com/sql/sql-server/> (дата звернення: 25.05.2026).

31. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. Geneva : ISO, 2011. 34 с.

32. Playwright Documentation / Microsoft. URL: <https://playwright.dev/docs/intro> (дата звернення: 26.05.2026).

33. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. 446 с.

ДОДАТКИ

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						85

Фрагменти програмного коду вебплатформи

У додатку наведено фрагменти програмного коду клієнтської (frontend) частини вебплатформи, реалізованої за допомогою фреймворку Vue.js [22].

Лістинг 2.1. Програмний код панелі адміністратора (AdminDashboard.vue)

```
const handleStatusChange = async (orderId: number, newStatus: string) => {
  try {
    await orderService.updateOrderStatus(orderId, newStatus);
    toast.success("Статус оновлено!");
    await loadSummary();
  } catch (error) {
    toast.error("Помилка оновлення");
    await loadOrders();
  }
};
```

Лістинг 2.2. Програмний код сторінки оформлення замовлення (Checkout.vue)

```
const submitOrder = async () => {
  if (cartStore.items.length === 0) return toast.error("Кошик порожній");
  if (!form.value.customerPhone.trim())
    return toast.warning("Вкажіть номер телефону");
  if (
    form.value.deliveryMethod !== "Pickup" &&
    !form.value.deliveryAddress.trim()
  ) {
    return toast.warning("Вкажіть адресу доставки");
  }
}
```

```
isSubmitting.value = true;
```

```
const orderData = {
  deliveryMethod: form.value.deliveryMethod,
  paymentMethod: form.value.paymentMethod,
  deliveryAddress:
    form.value.deliveryMethod === "Pickup"
```

Хіміч О.О.

Громський О.О.

```

        ? "СамоВивіз"
        : form.value.deliveryAddress,
customerPhone: form.value.customerPhone, // Відправляємо телефон
items: cartStore.items.map((item) => ({
    productId: item.id,
    quantity: item.quantity,
})),
});

try {
    await orderService.createOrder(orderData);
    cartStore.clearCart();
    toast.success("Замовлення успішно оформлено!");
    router.push("/profile");
} catch (error: any) {
    toast.error("Помилка при оформленні");
} finally {
    isSubmitting.value = false;
}
};

```

Лістинг 2.3. Програмний код панелі керування складом (InventoryPanel.vue)

```

<script setup lang="ts">
import { ref, onMounted } from "vue";
import { inventoryService } from "../services/inventoryService";
import { toast } from "vue3-toastify";

const report = ref<any[]>([]);
const isLoading = ref(true);

onMounted(async () => {
    try {
        report.value = await inventoryService.getReplenishmentReport();
    } catch (error) {
        toast.error("Не вдалося завантажити звіт із закупівель");
    } finally {

```

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						87

```

        isLoading.value = false;
    }
});

const printReport = () => {
    window.print();
};

const mockOrder = (name: string, qty: number) => {
    // Тут в ідеалі був би POST запит на створення документа закупівлі
    toast.success(`Сформовано запит постачальнику: ${name} (${qty} шт.)`);
};
</script>

```

Лістинг 2.4. Маршрутизація застосунку та розмежування доступу (router/index.ts)

```

import { createRouter, createWebHistory } from "vue-router";
import Home from "../pages/Home.vue";
import Login from "../pages/Login.vue";
import Register from "../pages/Register.vue";
import AdminDashboard from "../pages/AdminDashboard.vue";
import Checkout from "../pages/Checkout.vue";
import Profile from "../pages/Profile.vue";
import { authService } from "../services/authService";
import ProductDetails from "../pages/ProductDetails.vue";
import OrderDetails from "../pages/OrderDetails.vue";
import InventoryPanel from "../pages/InventoryPanel.vue";
import ForgotPassword from "../pages/ForgotPassword.vue";
import ResetPassword from "../pages/ResetPassword.vue";

const routes = [
    { path: "/", name: "Home", component: Home },
    { path: "/login", name: "Login", component: Login },
    { path: "/register", name: "Register", component: Register },
    { path: "/product/:id", name: "ProductDetails", component: ProductDetails },
    { path: "/order/:id", name: "OrderDetails", component: OrderDetails },
    {

```

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						88

```

    path: "/checkout",
    name: "Checkout",
    component: Checkout,
    meta: { requiresAuth: true },
  },
  {
    path: "/profile",
    name: "Profile",
    component: Profile,
    meta: { requiresAuth: true },
  },
  {
    path: "/admin",
    name: "AdminDashboard",
    component: AdminDashboard,
    meta: { requiresAuth: true, requiresAdmin: true },
  },
  {
    path: "/inventory",
    name: "InventoryPanel",
    component: InventoryPanel,
    meta: { requiresAuth: true, requiresAdmin: true },
  },
  {
    path: "/forgot-password",
    name: "ForgotPassword",
    component: ForgotPassword,
  },
  {
    path: "/reset-password",
    name: "ResetPassword",
    component: ResetPassword,
  },
];

```

```

const router = createRouter({
  history: createWebHistory(),
  routes,
});

router.beforeEach((to, from, next) => {
  const isAuthenticated = authService.isAuthenticated();
  const userRole = localStorage.getItem("user")
    ? JSON.parse(localStorage.getItem("user")).role
    : null;

  if (to.meta.requiresAuth && !isAuthenticated) {
    next("/login");
  } else if (to.meta.requiresAdmin && userRole !== "Admin") {
    next("/");
  } else if (
    to.meta.requiresWarehouse &&
    ![ "Admin", "WarehouseWorker" ].includes(userRole)
  ) {
    next("/");
  } else {
    next();
  }
});

export default router;

```

Лістинг 2.5. Модульний тест логіки кошика (cartStore.spec.ts)

```

import { describe, it, expect, beforeEach } from "vitest";
import { setActivePinia, createPinia } from "pinia";

describe("Cart Store (Unit Test)", () => {
  beforeEach(async () => {
    setActivePinia(createPinia());
    localStorage.clear();

```

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						90

```
const { cartStore } = await import("./cartStore");
cartStore.clearCart();
});
```

```
it("повинен правильно рахувати загальну суму товарів", async () => {
  const { cartStore } = await import("./cartStore");
  const product1 = { id: 1, name: "Чашка", price: 150.0, imageUrl: "", stockQuantity: 10 };
  const product2 = { id: 2, name: "Свічка", price: 200.0, imageUrl: "", stockQuantity: 10 };
  cartStore.addItem(product1);
  cartStore.addItem(product2);
  cartStore.addItem(product1);
  expect(cartStore.totalAmount).toBe(500.0);
  expect(cartStore.items.length).toBe(2);
  expect(cartStore.items[0].quantity).toBe(2);
});
```

```
it("повинен коректно видаляти товар та перераховувати суму", async () => {
  const { cartStore } = await import("./cartStore");
  const product1 = { id: 1, name: "Чашка", price: 150.0, imageUrl: "" };
  const product2 = { id: 2, name: "Свічка", price: 200.0, imageUrl: "" };
  cartStore.addItem(product1);
  cartStore.addItem(product2);
  cartStore.removeItem(1);
  expect(cartStore.items.length).toBe(1);
  expect(cartStore.items[0].id).toBe(2);
  expect(cartStore.totalAmount).toBe(200.0);
});
});
```

Лістинг 2.6. Тест сервісу автентифікації (authService.spec.ts)

```
import { describe, it, expect, vi, beforeEach } from "vitest";
import { authService } from "../authService";
import api from "../api/axiosConfig";
```

```
vi.mock("../api/axiosConfig", () => ({
  default: { post: vi.fn() },
```

Хіміч О.О.

Громський О.О.

ІПЗ.КР.Б-121-26-ПЗ

```
}});
```

```
describe("Auth Service (Integration Test)", () => {
```

```
  beforeEach(() => {  
    localStorage.clear();  
    vi.clearAllMocks();  
  });
```

```
  it("повинен зберігати токен у localStorage після успішного логіну", async () => {  
    const fakeResponse = { data: { token: "fake-jwt-token-123" } };  
    (api.post as any).mockResolvedValue(fakeResponse);  
    await authService.login({ email: "test@mail.com", password: "123" });  
    expect(api.post).toHaveBeenCalledWith("/auth/login", {  
      email: "test@mail.com",  
      password: "123",  
    });  
    expect(localStorage.getItem("token")).toBe("fake-jwt-token-123");  
  });
```

```
  it("повинен викидати помилку, якщо сервер повернув невірні дані", async () => {  
    const errorMessage = "Невірний пароль";  
    (api.post as any).mockRejectedValue(new Error(errorMessage));  
    await expect(  
      authService.login({ email: "test@mail.com", password: "wrong" }),  
    ).rejects.toThrow(errorMessage);  
    expect(localStorage.getItem("token")).toBeNull();  
  });  
});
```

Лістинг 2.7. Наскрізний тест користувацького сценарію (shop-flow.spec.ts)

```
import { test, expect } from "@playwright/test";
```

```
const APP_URL = "http://localhost:5173";
```

```
test("Повний цикл: логін та додавання товару в кошик", async ({ page }) => {  
  await page.goto(`${APP_URL}/login`);
```

		Хіміч О.О.			ІПЗ.КР.Б-121-26-ПЗ	
		Громський О.О.				
						92

```

await page.fill('input[type="email"]', "admin@giftshop.com");
await page.fill('input[type="password"]', "AdminPassword123!");
await page.click('button[type="submit"]');
await page.waitForTimeout(1000);

await expect(page).toHaveURL(new RegExp(`${APP_URL}/(profile)`));
await page.goto(`${APP_URL}/`);
await page.waitForTimeout(2000);

await page.getByTestId("add-to-cart-btn").first().click();
await page.waitForTimeout(2000);
const toastMessage = page.locator(".Toastify__toast-body");
await expect(toastMessage).toBeVisible();
});

```