Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра
зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Проєктування та реалізація інформаційної системи підтримки електронної комерції для малого бізнесу

Засвідчую, що в цій
кваліфікаційній роботі немає
запозичень із праць інших
авторів без відповідних посилань.

Студент гр. ІПЗ-22-2
_____ / Д. Д. Іващенко/

Керівник кваліфікаційної роботи _____ / А. В. Козиков /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2026

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-22-2 Іващенку Данилу Денисовичу

1. Тема: Проєктування та реалізація інформаційної системи підтримки електронної комерції для малого бізнесу

затверджено наказом по КНУ № 285с від «28» травня 2026 р.

2. Термін подання студентом закінченої роботи: «10» червня 2026 р.

3. Вихідні дані по роботі: розроблювана система повинна забезпечувати повний цикл електронної торгівлі – від каталогу товарів до оформлення та відстеження замовлень, реалізована засобами Python/Flask з використанням SQLite як СУБД.

4. Зміст пояснювальної записки: виконати аналіз існуючих рішень у сфері e-commerce для малого бізнесу; обґрунтувати вибір технологічного стеку; спроектувати архітектуру системи та структуру бази даних;

реалізувати програмне забезпечення; провести тестування та оцінку розробленої системи.

5. Перелік ілюстративного матеріалу: архітектурна діаграма системи, ER-діаграма бази даних, діаграма варіантів використання, схема маршрутизації Flask-додатку, знімки екранних форм.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання
1	Огляд літератури та аналіз існуючих рішень	25.02.2026 – 05.03.2026
2	Вибір технологічного стеку та обґрунтування архітектурних рішень	06.03.2026 – 15.03.2026
3	Проєктування бази даних та UML-діаграм	16.03.2026 – 25.03.2026
4	Підготовка матеріалів першого розділу	26.03.2026 – 02.04.2026
5	Реалізація серверної частини (Flask, SQLAlchemy)	03.04.2026 – 15.04.2026
6	Реалізація клієнтської частини (HTML-шаблони, Bootstrap)	16.04.2026 – 22.04.2026
7	Підготовка матеріалів другого та третього розділів	23.04.2026 – 12.05.2026
8	Тестування та виправлення помилок	13.05.2026 – 19.05.2026
9	Підготовка матеріалів четвертого розділу	20.05.2026 – 29.05.2026
10	Оформлення пояснювальної записки та презентації	30.05.2026 – 07.06.2026

Дата видачі завдання: «15» лютого 2026 р.

Студент _____ / Д. Д. Іващенко /

Керівник роботи _____ / А. В. Козиков /

РЕФЕРАТ

ЕЛЕКТРОННА КОМЕРЦІЯ, ІНФОРМАЦІЙНА СИСТЕМА, FLASK, SQLALCHEMY, SQLITE, REST-МАРШРУТИЗАЦІЯ, ПЛАТІЖНИЙ ШЛЮЗ, АДМІН-ПАНЕЛЬ, КОШИК ПОКУПЦЯ, АВТЕНТИФІКАЦІЯ.

Пояснювальна записка: 53 с., 9 табл., 23 рис., 3 дод., 18 джерел.

Метою кваліфікаційної роботи є проєктування та реалізація інформаційної системи підтримки електронної комерції для малого бізнесу на основі сучасного веб-стеку Python/Flask із реляційною базою даних.

У роботі проведено аналіз наявних платформ електронної торгівлі – WooCommerce, OpenCart, PrestaShop та конструкторів Shopify і Хорошоп – з погляду придатності для малого бізнесу з обмеженим IT-ресурсом. Виконано порівняння за критеріями вартості, масштабованості та технічної складності. Обґрунтовано вибір Python-мікрофреймворку Flask як основи для побудови власної системи.

Спроєктовано реляційну базу даних на п'яти сутностях: User, Product, Order, Review, Wishlist. Розроблено архітектуру у відповідності до шаблону MVC: серверна логіка відокремлена від HTML-шаблонів Jinja2. Реалізовано модуль автентифікації із хешуванням паролів (Werkzeug), рольовим розмежуванням доступу та механізмом відновлення пароля. Розроблено адмін-панель керування товарами та замовленнями, сесійний кошик покупця, систему відгуків, список бажань та заглушку платіжного шлюзу із симуляцією карткового платежу й оплати при доставці.

Проведено функціональне тестування всіх модулів системи. Результати тестування підтвердили коректність роботи системи відповідно до сформульованих вимог.

Розроблена система може бути використана як основа для інтернет-магазину малого бізнесу та як навчальний проєкт для вивчення технологій веб-розробки.

ABSTRACT

E-COMMERCE, INFORMATION SYSTEM, FLASK, SQLALCHEMY, SQLITE, REST ROUTING, PAYMENT GATEWAY, ADMIN PANEL, SHOPPING CART, AUTHENTICATION.

Thesis in 53 p., 9 tab., 23 fig., 3 app., 18 references.

The goal of the thesis is to design and implement an information system for e-commerce support for small businesses based on the modern Python/Flask web stack with a relational database.

Existing e-commerce platforms – WooCommerce, OpenCart, PrestaShop, Shopify, and Khoroshop – were analyzed from the perspective of suitability for small businesses with limited IT resources. A comparison was made by cost, scalability, and technical complexity. The choice of the Python microframework Flask as the foundation for building a custom system was justified.

A relational database with five entities was designed: User, Product, Order, Review, and Wishlist. The architecture was developed according to the MVC pattern. An authentication module with password hashing (Werkzeug), role-based access control, and password recovery mechanism was implemented. An admin panel for product and order management, a session-based shopping cart, a review system, a wishlist, and a payment gateway stub simulating card payment and cash-on-delivery were developed.

Functional testing of all system modules was performed. The test results confirmed the correct operation of the system in accordance with the defined requirements.

ЗМІСТ

ВСТУП

Стрімкий розвиток цифрових технологій та зміна споживчої поведінки зумовили масовий перехід бізнесу у простір електронної комерції. За даними Державної служби статистики України, обсяг роздрібної інтернет-торгівлі щороку зростає на 20–35 %, а кількість малих підприємств, що розглядають онлайн-канал збуту як пріоритетний, перевищила 60 % від загальної кількості суб'єктів малого підприємництва [18]. Водночас значна частина малих бізнесів стикається з проблемою відсутності доступного, гнучкого та функціонально повного рішення для організації власного інтернет-магазину без залежності від сторонніх платформ із дорогим передплатним планом.

Актуальність теми обумовлена потребою у створенні відкритого програмного рішення, яке дозволяє малому бізнесу самостійно розгортати інтернет-магазин з повним контролем над функціональністю, структурою даних і бізнес-логікою. Використання Python-мікрофреймворку Flask як основи забезпечує мінімалізм і гнучкість архітектурних рішень, тоді як SQLAlchemy надає зручний ORM-доступ до реляційної бази даних SQLite без необхідності розгортання окремого сервера СУБД.

Метою кваліфікаційної роботи є проєктування та програмна реалізація інформаційної системи підтримки електронної комерції, що охоплює повний цикл взаємодії покупця та продавця: перегляд каталогу, пошук, оформлення замовлення, оплату, сповіщення та адміністрування.

Для досягнення мети поставлено такі завдання:

– проаналізувати існуючі платформи та рішення у сфері e-commerce для малого бізнесу;

– обґрунтувати вибір технологічного стеку та архітектурних підходів;

– спроєктувати реляційну базу даних і UML-діаграми системи;

– реалізувати серверну та клієнтську частини інформаційної системи;

– провести функціональне тестування розроблених модулів;

– оцінити результати та визначити напрями подальшого розвитку системи.

Об'єктом розробки є веб-застосунок електронної комерції для малого бізнесу з повним циклом обробки замовлень. Предметом розробки є методи та інструменти проєктування і реалізації інформаційних систем підтримки e-commerce з використанням мікрофреймворку Flask.

Практична значущість роботи полягає у розробці функціонально повного прототипу інтернет-магазину, який може бути розгорнутий на власному сервері та адаптований до специфіки конкретного малого бізнесу без прив'язки до комерційних платформ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз ринку рішень для електронної комерції

Електронна комерція (e-commerce) охоплює всі комерційні транзакції, що здійснюються через мережу Інтернет: купівлю-продаж товарів і послуг, електронні платежі, управління замовленнями та взаємодію з клієнтами в онлайн-просторі. Для малого бізнесу вибір відповідної платформи або рішення для організації інтернет-магазину є стратегічним рішенням, що впливає на витрати, гнучкість та масштабованість бізнесу.

Ринок рішень для e-commerce поділяється на кілька ключових сегментів:

– SaaS-платформи (Software-as-a-Service) – хмарні рішення з передплатою: Shopify, BigCommerce, Squarespace, Хорошоп (Prom.ua). Ці платформи надають готовий функціонал без необхідності технічних знань, але обмежують кастомізацію та передбачають відсоткову комісію з продажів.

– Open-source CMS/e-commerce – самостійно розгорнуті рішення з відкритим кодом: WooCommerce (WordPress), OpenCart, PrestaShop, Magento. Забезпечують максимальну гнучкість, але потребують хостингу, технічного обслуговування та IT-компетенцій.

– Кастомна розробка – власне програмне забезпечення, розроблене на замовлення або за допомогою фреймворків. Забезпечує повну відповідність бізнес-вимогам, але потребує значних ресурсів на розробку.

– Marketplace-рішення – присутність на загальних торгових майданчиках (Rozetka, Prom.ua, OLX). Мінімальний поріг входу, але відсутність власного брендингу та залежність від правил платформи.

За даними аналітиків BuiltWith та SimilarTech, станом на 2025 рік розподіл технологічних платформ серед українських інтернет-магазинів виглядає наступним чином (таблиця 1.1) [14].

Таблиця 1.1 - Розподіл технологічних платформ

Платформа	Частка ринку, %	Основна аудиторія
WooCommerce	31	Малий та середній бізнес
OpenCart	18	Малий бізнес
PrestaShop	12	Малий та середній бізнес
Shopify	11	Малий бізнес (SaaS)
Хорошоп	9	Малий бізнес (SaaS, Україна)
Кастомна розробка	8	Середній та великий бізнес
Інші	11	Різні сегменти

Аналіз ринкових даних свідчить, що найпоширенішими рішеннями для малого бізнесу залишаються WooCommerce та OpenCart завдяки безкоштовній ліцензії та великій спільноті. Однак обидві платформи несуть приховані витрати на хостинг, преміум-розширення та технічне обслуговування.

1.2 Огляд існуючих платформ та їх порівняльний аналіз

Для обґрунтування вибору підходу до розробки виконано детальний аналіз п'яти найбільш релевантних для малого бізнесу платформ і рішень.

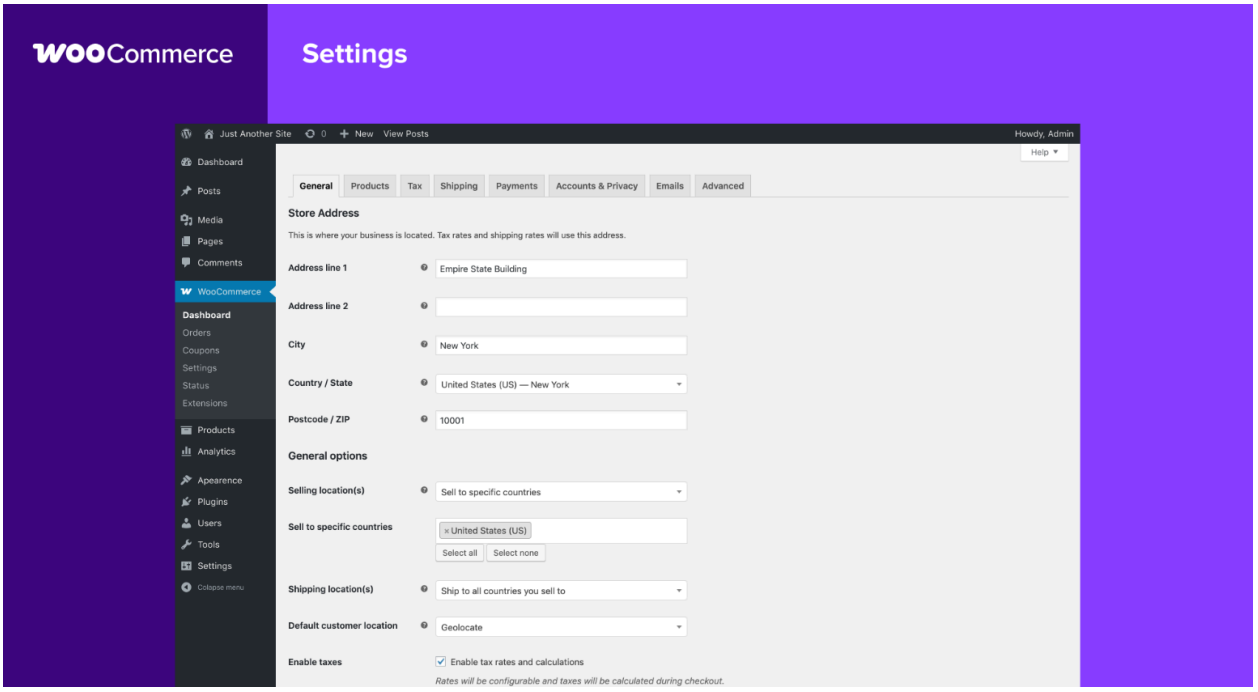
1.2.1 WooCommerce

WooCommerce є безкоштовним плагіном для системи управління вмістом WordPress і займає лідируючу позицію на глобальному ринку e-commerce з часткою понад 27 % всіх інтернет-магазинів світу [13]. Архітектура системи побудована на PHP та базується на механізмі подій і

фільтрів WordPress (hooks). База даних MySQL зберігає товари, замовлення та налаштування у стандартизованій структурі.

Переваги WooCommerce: гнучка система розширень (понад 800 офіційних плагінів), інтеграція з усіма основними платіжними системами, великий вибір тем оформлення, активна спільнота розробників. Недоліки: висока початкова складність для не технічних користувачів, значні витрати на преміум-розширення (від $50 до $300 за плагін), необхідність самостійного адміністрування WordPress, регулярні питання безпеки.

Приклад роботи WooCommerce (*див. рис. 1.1*)



1.2.2 OpenCart

OpenCart – безкоштовна CMS з відкритим кодом, написана на PHP. Система розроблена спеціально для електронної комерції та надає повноцінний функціонал без необхідності встановлення додаткових плагінів. Архітектура заснована на патерні MVC (Model-View-Controller).

OpenCart має зрозумілий адмін-інтерфейс, вбудовану підтримку кількох мов і валют. Однак система має застарілу кодову базу (PHP 7.x), обмежену

підтримку сучасних стандартів програмування та проблеми з продуктивністю при великому каталозі.

Приклад як виглядає OpenCart (*див рис. 1.2*)

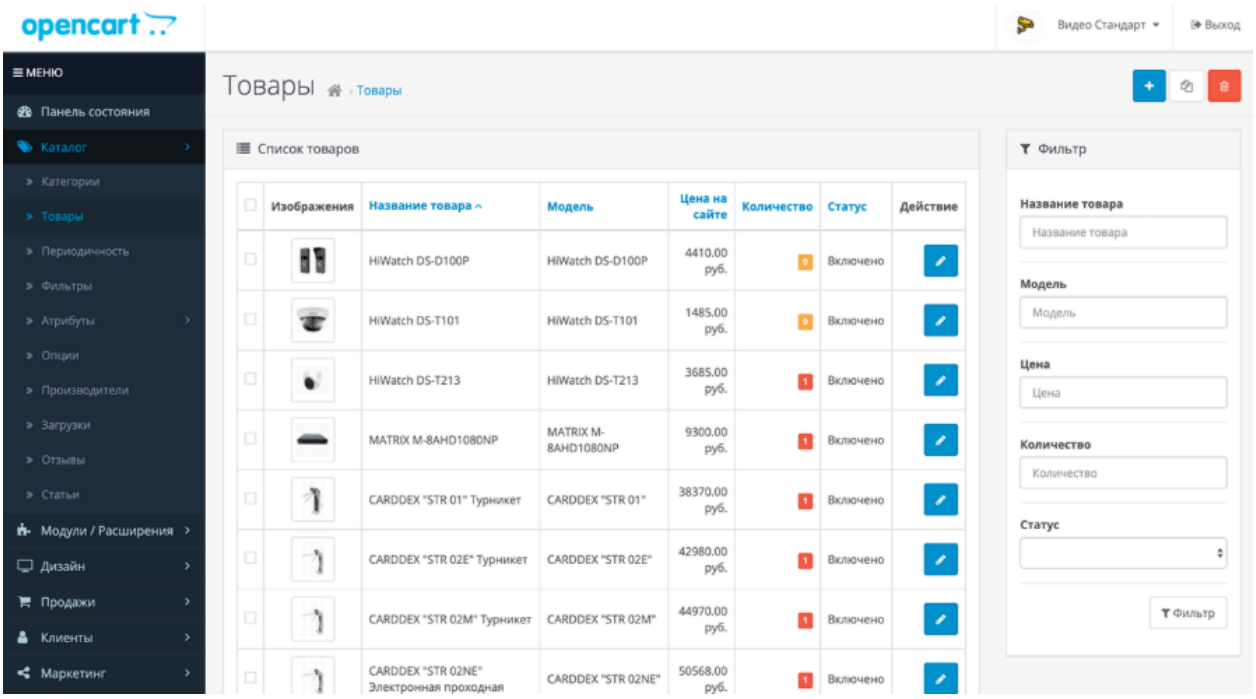


Рисунок 1.2 – OpenCart

1.2.3 PrestaShop

PrestaShop – open-source e-commerce платформа, орієнтована на середній сегмент ринку. Написана на PHP, використовує Symfony-компоненти у версіях 1.7+. Надає потужний функціонал для управління складними каталогами, акційними правилами та логістикою.

Приклад як виглядає PrestaShop (*див. рис. 1.3)*

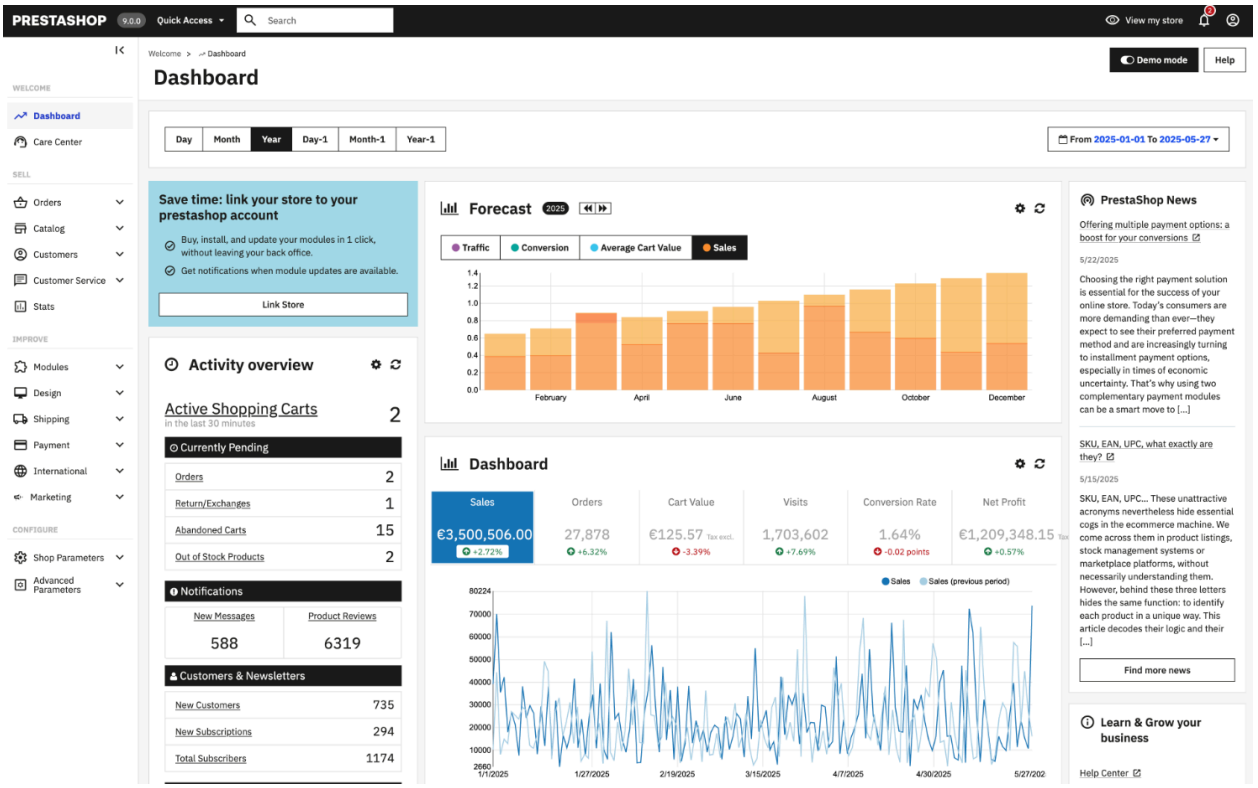


Рисунок 1.3 – PrestaShop

1.2.4 Shopify

Shopify – провідна SaaS-платформа, яка дозволяє запустити інтернет-магазин без технічних знань [13]. Вартість тарифів: Basic ($29/міс.), Shopify ($79/міс.), Advanced ($299/міс.), плюс транзакційна комісія від 0,5 % до 2 %. Для малого бізнесу з обмеженим бюджетом щорічні витрати можуть становити від $500 до $3600, що є суттєвим навантаженням.

Приклад як виглядає Shopify (*див. рис. 1.4)*

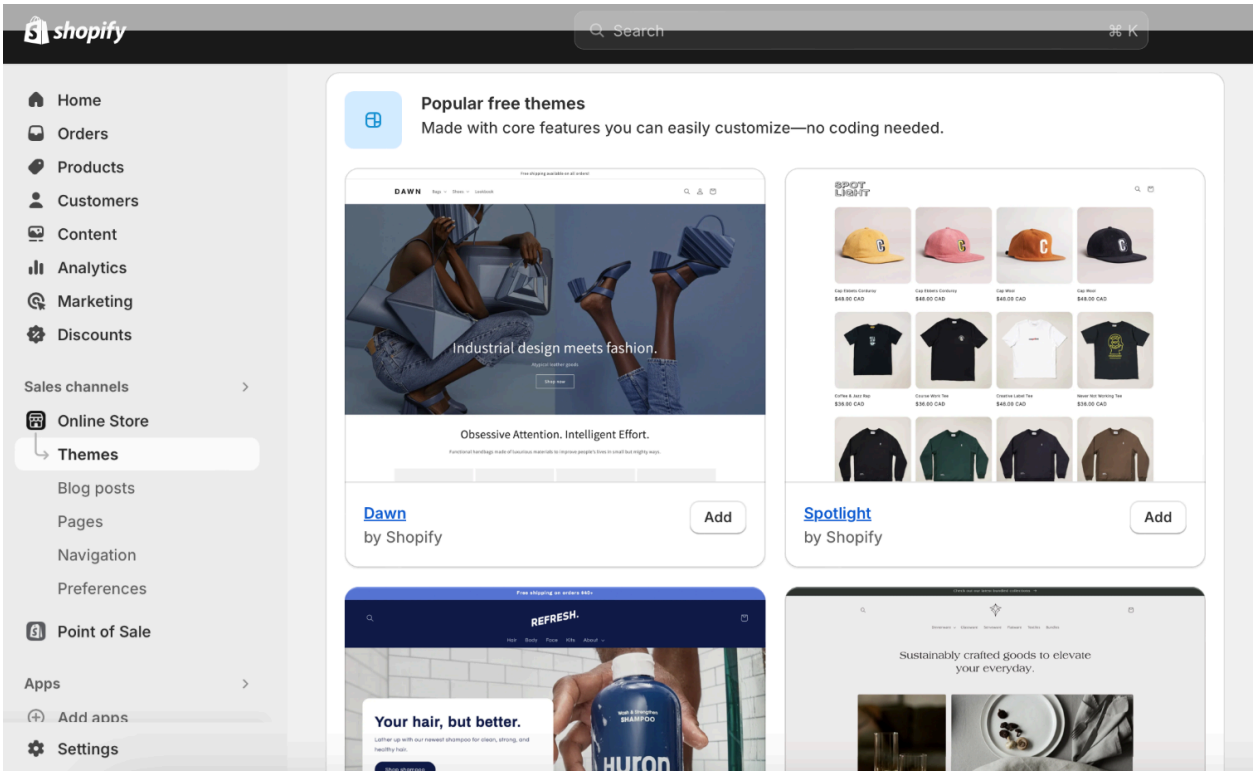


Рисунок 1.4 – Платформа Shopify

1.2.5 Хорошоп

Хорошоп – українська SaaS-платформа від компанії Prom.ua, адаптована до українського ринку. Інтегрована з Nova Poshta, Укрпоштою, ПриватБанком та monobank. Тарифи від 599 грн/міс. Перевага – повна локалізація та інтеграція з українськими сервісами. Обмеження – відсутність повного контролю над кодом та залежність від провайдера.

Приклад як виглядає платформа Хорошоп (*див. рис. 1.5)*

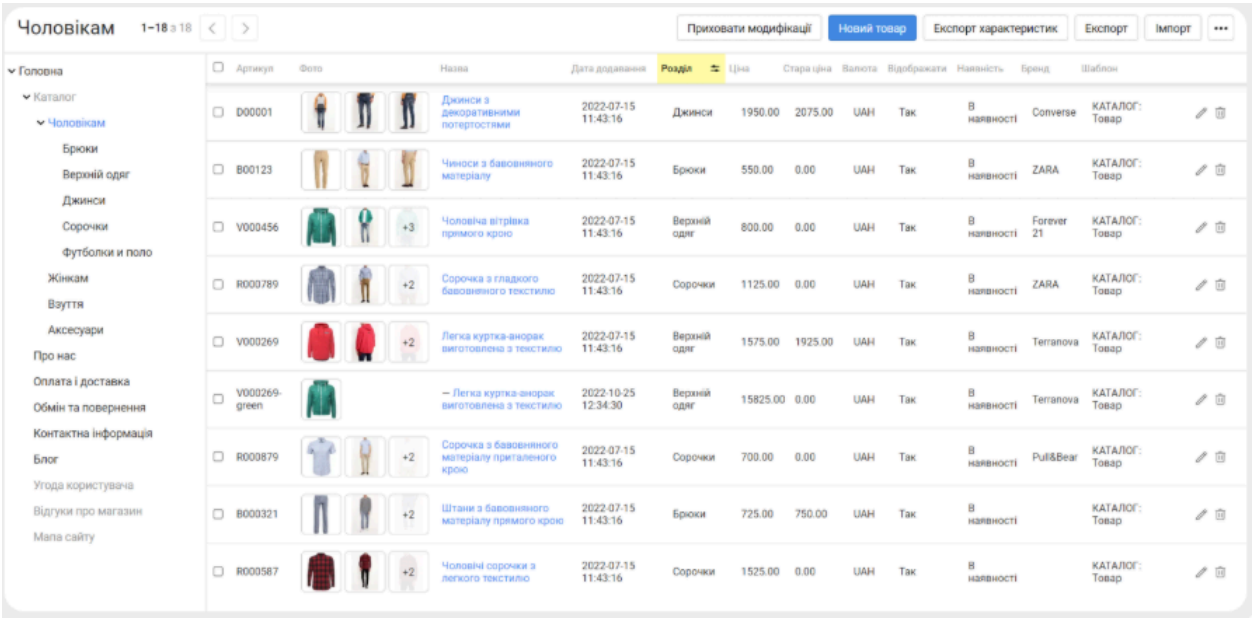


Рисунок 1.5 – Платформа Хорошоп

Таблиця 1.2 - Порівняльний аналіз розглянутих рішень

Критерій	WooCommerce	OpenCart	Shopify	Хорошоп	Flask (власна)
Вартість	Безкоштовно + хостинг + плагіни	Безкоштовно + хостинг	$29–299/міс.	від 599 грн/міс.	Безкоштовно + хостинг
Гнучкість	Висока	Середня	Низька	Низька	Максимальна
Технічна складність	Середня	Середня	Низька	Низька	Висока
Контроль над кодом	Так	Так	Ні	Ні	Повний
Локалізація (UA)	Часткова	Часткова	Часткова	Повна	Вільна
Масштабованість	Середня	Середня	Висока	Середня	Висока

З аналізу випливає, що для малого бізнесу з технічними ресурсами або ІТ-фахівцем у команді власна розробка на Flask забезпечує найвищу гнучкість і найнижчу довгострокову вартість утримання при максимальному контролі над функціональністю.

1.3 Постановка задачі та вимоги до системи

На підставі проведеного аналізу сформульовано задачу: розробити інформаційну систему підтримки електронної комерції для малого бізнесу, яка відповідає наведеним нижче функціональним та нефункціональним вимогам.

Функціональні вимоги:

– каталог товарів із категоріями, пошуком і пагінацією (не менше 6 товарів на сторінку);

– сторінка деталей товару з описом, ціною, залишком та відгуками;

– сесійний кошик покупця з можливістю додавання, видалення та очищення;

– форма оформлення замовлення з підтримкою двох методів оплати (картка, накладений платіж);

– симуляція платіжного шлюзу з відповідями "успіх" і "помилка";

– email-підтвердження замовлення у форматі HTML;

– реєстрація та авторизація користувача з хешуванням пароля;

– відновлення пароля через секретне запитання;

– список бажань (Wishlist) для авторизованих користувачів;

– система відгуків із рейтингом 1–5 зірок;

– особистий кабінет із переліком замовлень;

– адмін-панель із двоетапною верифікацією та управлінням товарами й замовленнями.

Нефункціональні вимоги:

– безпека: паролі та секретні відповіді зберігаються у хешованому вигляді (PBKDF2-HMAC-SHA256);

– конфіденційні параметри (SECRET_KEY, SMTP-дані) зберігаються у .env-файлі;

– час відгуку системи при запитах до каталогу не повинен перевищувати 2 секунди;

– система повинна підтримувати одночасну роботу до 50 активних сесій;

– зручність адміністрування без необхідності правки вихідного коду.

Таблиця 1.3 – Варіанти використання системи

Варіант використання	**Актор**	**Опис**
UC-01 Перегляд каталогу	Відвідувач	Пошук і фільтрація товарів
UC-02 Додавання до кошика	Відвідувач	Додавання товару до сесійного кошика
UC-03 Оформлення замовлення	Відвідувач	Введення даних та вибір оплати
UC-04 Реєстрація	Відвідувач	Створення облікового запису
UC-05 Авторизація	Користувач	Вхід до системи
UC-06 Перегляд кабінету	Користувач	Замовлення та список бажань
UC-07 Залишення відгуку	Користувач	Оцінка та коментар до товару
UC-08 Управління товарами	Адміністратор	Додавання, редагування товарів
UC-09 Управління замовленнями	Адміністратор	Перегляд та зміна статусів

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір технологічного стеку

Вибір технологічного стеку є фундаментальним архітектурним рішенням, що визначає продуктивність, масштабованість та зручність супроводу системи. Враховуючи сформульовані вимоги, проаналізовано кілька альтернативних варіантів.

Розглянуті бекенд-фреймворки:

Таблиця 2.1 – Порівняння бекенд-фреймворків

Фреймворк	Мова	Тип	ORM	Особливості
Flask	Python	Мікрофрейм ворк	SQLAlchemy	Мінімалізм, гнучкість, легке розгортання
Django	Python	Full-stack	Django ORM	Готові рішення, "батареї в комплекті"
FastAPI	Python	API-фреймв орк	SQLAlchemy	Асинхронність, автодокументація OpenAPI
Express.js	JavaScript	Мікрофрейм ворк	Sequelize	Швидкодія, Node.js екосистема
Laravel	PHP	Full-stack	Eloquent ORM	Елегантний синтаксис, Blade-шаблони

Для даної кваліфікаційної роботи обрано стек Python/Flask [3] з таких міркувань:

– Flask дотримується принципу "роби лише те, що потрібно" – мінімальний набір вбудованих компонентів дозволяє точно контролювати залежності та структуру проєкту [6];

– SQLAlchemy забезпечує потужний ORM з підтримкою міграцій та гнучким query API [7];

– SQLite не потребує окремого сервера СУБД – ідеально для прототипування та малого навантаження;

– Jinja2 (вбудований шаблонний рушій Flask) підтримує спадкування шаблонів і макроси [9];

– Flask-Mail надає простий інтерфейс для надсилання email через SMTP [6];

– Werkzeug забезпечує безпечне хешування паролів за алгоритмом PBKDF2-HMAC-SHA256 [8].

Обрані технології та їх версії наведено у таблиці 2.2.

Таблиця 2.2 – Технологічний стек проекту

Технологія	Версія	Призначення
Python	3.11	Основна мова програмування
Flask	3.0.0	Веб-фреймворк
Flask-SQLAlchemy	3.1.1	ORM-інтеграція
Flask-Mail	0.9.1	Надсилання email
Werkzeug	3.0.1	Хешування паролів, HTTP-утиліти
python-dotenv	1.0.0	Управління конфігурацією з .env
SQLite	3.x	Реляційна СУБД
Jinja2	3.1.x	Шаблонний рушій HTML
Bootstrap	5.3	CSS-фреймворк (клієнтська частина)

2.2 Архітектура системи

Система побудована за архітектурним шаблоном MVC (Model-View-Controller) [15], адаптованим для Flask:

– Model (Модель) – класи SQLAlchemy: User, Product, Order, Review, Wishlist. Відповідають за зберігання та бізнес-логіку даних.

– View (Подання) – Jinja2-шаблони в директорії templates/. Відповідають за відображення даних у браузері.

– Controller (Контролер) – маршрути Flask (route-функції) у файлі app.py. Обробляють HTTP-запити, взаємодіють з моделями та передають дані у шаблони.

Додатково виокремлено сервісний шар у вигляді класу PaymentGateway та функції send_order_confirmation_email, що інкапсулюють логіку оплати та Email-взаємодії.

Архітектура системи складається з таких рівнів:

– Рівень представлення (Presentation Layer): HTML-шаблони на основі Jinja2 та Bootstrap 5 для адаптивного відображення на різних пристроях.

– Рівень логіки додатку (Application Layer): Flask-маршрути, декоратори доступу (login_required, admin_required), обробка форм і сесій.

– Рівень даних (Data Layer): SQLAlchemy ORM поверх SQLite-бази даних, що забезпечує абстракцію від SQL-запитів.

– Зовнішні сервіси: SMTP-сервер для email-сповіщень, платіжний шлюз (stub-реалізація).

Взаємодія компонентів системи при обробці запиту оформлення замовлення включає наступну послідовність операцій:

– 1) Браузер надсилає POST-запит на /checkout з даними форми.

– 2) Декоратор сесії перевіряє наявність кошика.

– 3) Маршрут checkout() зчитує дані форми, перевіряє наявність товарів на складі.

– 4) PaymentGateway.process_card_payment() або process_cod_payment() обробляє оплату.

– 5) Новий об'єкт Order зберігається у базі даних; залишки товарів зменшуються.

– 6) send_order_confirmation_email() надсилає HTML-лист покупцю.

– 7) Кошик очищується з сесії; браузер отримує сторінку підтвердження.

2.3 Проєктування бази даних

База даних системи містить п'ять основних сутностей. Нижче наведено їх структуру та зв'язки.

Сутність User (Користувач):

Таблиця 2.3 – Структура таблиці User

Поле	Тип	Обмеження	Опис
id	INTEGER	PK, AUTO	Унікальний ідентифікатор
username	VARCHAR(50)	UNIQUE, NOT NULL	Ім'я користувача
email	VARCHAR(100)	UNIQUE, NOT NULL	Електронна адреса
password_hash	VARCHAR(200)	NOT NULL	Хеш пароля (PBKDF2)
is_admin	BOOLEAN	DEFAULT FALSE	Роль адміністратора
secret_question	VARCHAR(150)	NOT NULL	Секретне запитання
secret_answer	VARCHAR(100)	NOT NULL	Хеш відповіді

Сутність Product (Товар): id, name (VARCHAR 100), price (FLOAT), description (TEXT), stock_quantity (INTEGER, DEFAULT 10), image_url (VARCHAR 255), category (VARCHAR 100), subcategory (VARCHAR 100).

Сутність Order (Замовлення): id, user_id (FK → User), user_name, user_email, user_address, total_amount (FLOAT), items (TEXT – JSON масив), status (VARCHAR 50, DEFAULT "Нове"), order_date (DATETIME), payment_method, payment_status.

Сутність Review (Відгук): id, product_id (FK → Product, CASCADE DELETE), user_id (FK → User), rating (INTEGER 1–5), text (TEXT), date_posted (DATETIME).

Сутність Wishlist (Список бажань): id, user_id (FK → User), product_id (FK → Product).

Зв'язки між сутностями:

– User : Order = 1 : N (один користувач може мати багато замовлень);

– User : Review = 1 : N (один користувач може залишати багато відгуків);

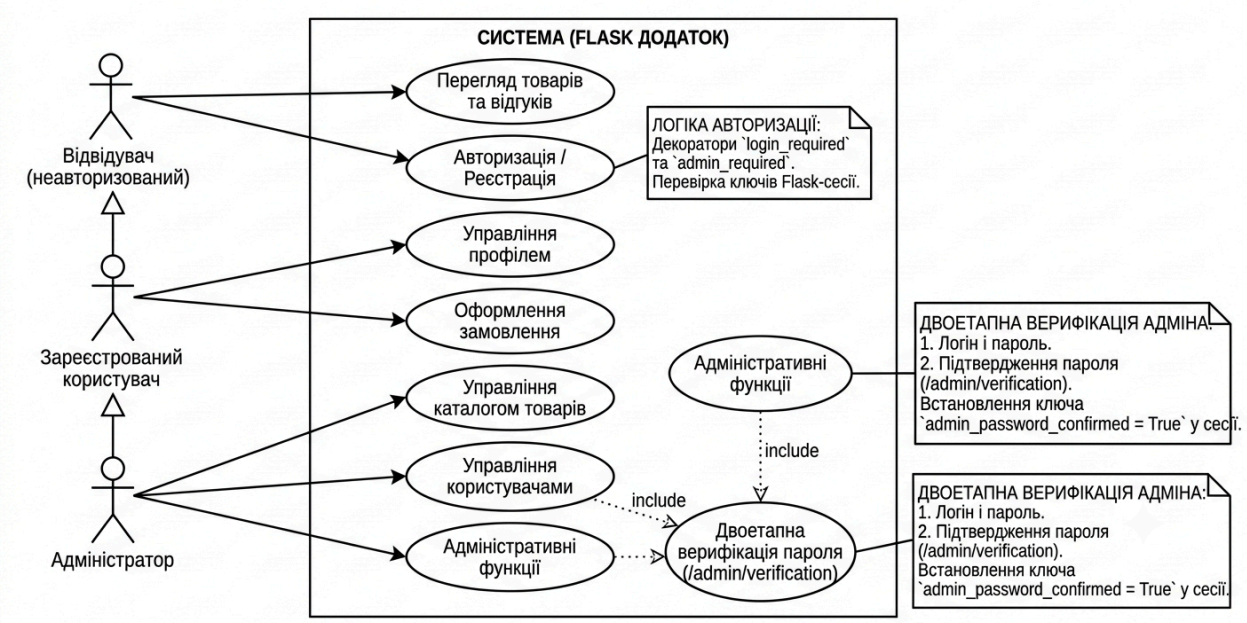
– User : Wishlist = 1 : N (один користувач може мати багато позицій у списку бажань);

– Product : Review = 1 : N з CASCADE DELETE (видалення товару видаляє всі відгуки);

– Product : Wishlist = 1 : N (один товар може бути в списку бажань кількох користувачів).

2.4 Діаграма варіантів використання

2.4 Діаграма варіантів використання Системи (Flask)



Система підтримує три ролі акторів: Відвідувач (неавторизований), Зареєстрований користувач та Адміністратор. Кожна роль має доступ до певного набору варіантів використання, визначених у розділі 1.3. Авторизація реалізована через декоратори login_required та admin_required, що перевіряють наявність відповідних ключів у Flask-сесії. Адміністратор проходить двоетапну верифікацію: перший вхід із логіном і паролем, потім підтвердження пароля на сторінці /admin/verification. Після підтвердження у сесії встановлюється ключ admin_password_confirmed = True, який перевіряється при кожному зверненні до захищених адмін-маршрутів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Структура проєкту

Проєкт організовано у відповідності до стандартних конвенцій Flask-додатків. Структура директорій:

```
ecommerce_flask_project/
├── app.py                    # Головний модуль:
моделі, маршрути, конфігурація
├── requirements.txt          # Залежності Python
├── .env                      # Конфігураційні змінні
(не в git)
├── .gitignore
├── ecommerce.db              # SQLite база даних
(генерується при запуску)
├── static/
│   └── css/
│       └── style.css         # Кастомні стилі
└── templates/
    ├── base.html             # Базовий шаблон
(навігація, footer)
    ├── catalog.html          # Каталог товарів
    ├── product_detail.html # Деталі товару + відгуки
    ├── cart.html             # Кошик
    ├── checkout_success.html  # Підтвердження
замовлення
    ├── login.html            # Авторизація
    ├── register.html         # Реєстрація
    ├── forgot_password.html
    ├── my_orders.html        # Особистий кабінет
    ├── admin_login.html      # Sudo-верифікація адміна
    ├── admin_dashboard.html
    ├── admin_product_form.html
    └── info.html
```

Файл app.py логічно розподілено на блоки за допомогою коментарів-роздільників: конфігурація Flask, моделі SQLAlchemy, платіжний шлюз, допоміжні функції, декоратори, контекстний процесор, функція seed_database та маршрути.

3.2 Реалізація моделей бази даних

Всі моделі успадковують клас db.Model бібліотеки Flask-SQLAlchemy. Зв'язки між таблицями оголошуються через db.relationship() з параметром backref для зворотного доступу. Каскадне видалення відгуків при видаленні товару реалізовано параметром cascade="all, delete-orphan".

Нижче наведено ключові фрагменти реалізації моделей:

```
class User(db.Model):
    id              = db.Column(db.Integer,
primary_key=True)
    username        = db.Column(db.String(50),
unique=True, nullable=False)
    email           = db.Column(db.String(100),
unique=True, nullable=False)
    password_hash   = db.Column(db.String(200),
nullable=False)
    is_admin        = db.Column(db.Boolean,
default=False)
    secret_question = db.Column(db.String(150),
nullable=False)
    secret_answer   = db.Column(db.String(100),
nullable=False)
    orders   = db.relationship("Order",
backref="buyer", lazy=True)
    wishlist = db.relationship("Wishlist",
backref="user",  lazy=True)
    reviews  = db.relationship("Review",
backref="author", lazy=True)
```

Для безпечного зберігання паролів використовується функція generate_password_hash() бібліотеки Werkzeug [8], яка застосовує алгоритм pbkdf2:sha256 із сіллю (salt) довжиною 16 байт та 260000 ітераціями хешування. Перевірка паролю виконується через check_password_hash(), що унеможливлює атаку за допомогою попередньо обрахованих таблиць (rainbow tables).

Аналогічний механізм хешування застосовано до секретних відповідей у функції відновлення пароля, що забезпечує захист навіть у разі витоку бази даних.

3.3 Реалізація серверної логіки (маршрути)

Серверна логіка системи організована у вигляді Flask-маршрутів, що відповідають принципам REST-архітектури. Маршрути об'єднано у функціональні групи.

Маршрут каталогу підтримує одночасну фільтрацію за трьома параметрами та пагінацію:

```
@app.route("/catalog")
def catalog():
    category     = request.args.get("category",
"").strip()
    subcategory  = request.args.get("subcategory",
"").strip()
    search_query = request.args.get("q",
"").strip()
    page         = request.args.get("page", 1, type=int)
    query = Product.query
    if category:
        query = query.filter_by(category=category)
    if subcategory:
        query = query.filter_by(subcategory=subcategory)
    if search_query:
        pattern = f"%{search_query}%"
        query = query.filter(
            Product.name.ilike(pattern) |
Product.description.ilike(pattern)
        )
    pagination = query.paginate(page=page, per_page=6,
error_out=False)
```

Маршрут оформлення замовлення реалізує транзакційну логіку з rollback при помилці:

```
@app.route("/checkout", methods=["POST"])
def checkout():
    try:
        # 1. Перевірка наявності товарів
        # 2. Обробка платежу через PaymentGateway
        # 3. Зменшення stock_quantity
        # 4. Збереження Order у БД
        # 5. Надсилання email
        # 6. Очищення кошика
    except Exception as e:
        db.session.rollback()
        flash(f"Помилка: {e}", "danger")
```

Декоратори доступу реалізовані через functools.wraps для збереження метаданих функцій:

```
def login_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if "user_id" not in session:
            flash("Будь ласка, увійдіть у свій акаунт.",
"warning")
            return redirect(url_for("login"))
        return f(*args, **kwargs)
    return decorated
```

Таблиця маршрутів системи наведена нижче. (див. табл. 3.1)

Таблиця 3.1 – Маршрути Flask-додатку

URL	Метод	Функція	Опис	Доступ
/	GET	index()	Редирект на каталог	Публічний
/catalog	GET	catalog()	Каталог з пошуком	Публічний
/product/<id>	GET	product_detail()	Деталі товару	Публічний

URL	Метод	Функція	Опис	Доступ
/product/<id>/review	POST	add_review()	Додати відгук	Авторизований
/register	GET/POST	register()	Реєстрація	Публічний
/login	GET/POST	login()	Авторизація	Публічний
/logout	GET	logout()	Вихід	Авторизований
/forgot-password	GET/POST	forgot_password()	Відновлення пароля	Публічний
/add_to_cart/<id>	GET	add_to_cart()	Додати до кошика	Публічний
/cart	GET	cart()	Кошик	Публічний
/checkout	POST	checkout()	Оформлення	Публічний
/my-account	GET	my_orders()	Особистий кабінет	Авторизований
/wishlist/add/<id>	GET	add_to_wishlist()	Додати до Wishlist	Авторизований
/admin	GET	admin_dashboard()	Адмін-панель	Адміністратор
/admin/product/add	POST	admin_add_product()	Додати товар	Адміністратор
/admin/order/<id>/status	POST	admin_order_status()	Змінити статус	Адміністратор

3.4 Клієнтська частина (шаблони)

Клієнтська частина побудована на базовому шаблоні base.html, що містить навігаційну панель Bootstrap 5 [5], глобальні Flash-повідомлення та блоки content і scripts для перевизначення у дочірніх шаблонах. Механізм спадкування шаблонів Jinja2 забезпечує відсутність дублювання HTML-коду.

Навігаційна панель динамічно відображає посилання залежно від стану авторизації: для неавторизованих – посилання "Увійти" та "Реєстрація", для авторизованих – "Мій кабінет" та "Вихід", для адміністратора – додатково

"Адмін-панель". Лічильник кошика відображається як бейдж над іконкою та оновлюється через змінну cart_count, що передається у шаблони через context_processor.

Каталог товарів реалізовано у вигляді сітки карток Bootstrap (row g-4), де кожна картка містить фотографію товару, назву, ціну, категорію, кнопки "Купити" та "В обране". Кольорове позначення: якщо товар є у списку бажань, кнопка серця стає активною (text-danger).

Форма оформлення замовлення у шаблоні cart.html розроблена у два блоки: перелік товарів кошика з підсумком та форма введення даних покупця. Вибір методу оплати реалізовано через радіо-кнопки Bootstrap з умовним відображенням поля введення токена картки через JavaScript.

Сторінка деталей товару product_detail.html відображає повний опис, залишок на складі (з попередженням при stock <= 3), середній рейтинг на основі всіх відгуків та форму для залишення нового відгуку. Зірковий рейтинг реалізовано засобами HTML + CSS без зовнішніх бібліотек.

3.5 Платіжний шлюз та підтвердження замовлень

Клас PaymentGateway реалізований як stub (заглушка) для демонстрації інтеграції з реальними платіжними сервісами. Архітектурне рішення – статичні методи класу без стану – дозволяє легко замінити реалізацію на виклики реального API (LiqPay, WayForPay, Stripe) без зміни коду маршруту checkout().

```
class PaymentGateway:
    @staticmethod
    def process_card_payment(amount: float,
card_token: str) -> dict:
        if card_token == "fail_test":
            return {"success": False, "error":
"Платіж відхилено банком."}
        # У реальній системі: виклик API
LiqPay/WayForPay/Stripe
```

```
            import random
            txn_id = f"TXN-{random.randint(100000,
    999999)}"
            return {"success": True, "transaction_id":
    txn_id}
```

Токен "fail_test" є спеціальним тестовим значенням для перевірки сценарію відмови платежу. При реальній інтеграції токен генерується на стороні клієнта платіжним SDK (наприклад, Fondy Checkout або LiqPay JS SDK) і передається на сервер через захищене HTTPS-з'єднання.

HTML-лист підтвердження замовлення формується у функції send_order_confirmation_email() методом конкатенації рядків Python. Лист містить: шапку з логотипом магазину, таблицю з переліком товарів та цінами, підсумкову суму, адресу доставки, статус оплати та контактну інформацію підтримки. Відправка здійснюється через Flask-Mail з SMTP-налаштуваннями із .env-файлу; помилка SMTP логується, але не зупиняє процес оформлення замовлення.

3.6 Адмін-панель

Адміністративний інтерфейс системи доступний за маршрутом /admin та потребує проходження двоетапної верифікації. Після успішного входу адміністратор бачить три розділи: управління товарами, активні замовлення та архів завершених замовлень.

Управління товарами: форма додавання нового товару з полями назва, ціна, категорія, підкатегорія, кількість на складі, URL фото та опис. Редагування існуючого товару відкривається у модальному вікні Bootstrap з попередньо заповненими полями. Всі операції виконуються через POST-запити до відповідних маршрутів.

Управління замовленнями: адміністратор бачить перелік усіх активних замовлень з деталями (ім'я покупця, адреса, перелік товарів, сума, метод оплати, дата). Для кожного замовлення доступна форма зміни статусу:

"Нове", "В обробці", "Відправлено", "Виконано", "Скасовано". Архів відображає завершені та скасовані замовлення окремою таблицею.

Перелік товарів відображається у таблиці з колонками: ID, назва, категорія, ціна, залишок, дії (редагувати). Рядки з залишком менше 3 одиниць виділяються попередженням для привернення уваги адміністратора.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Стратегія тестування

Тестування розробленої інформаційної системи проводилось на рівні функціонального тестування "чорної скриньки" (black-box testing), яке перевіряє відповідність поведінки системи специфікованим вимогам без врахування внутрішньої структури коду [12]. Для кожного варіанту використання (UC-01 – UC-09) розроблено тест-кейси, що охоплюють позитивні та негативні сценарії.

Принципи, яких дотримувались при тестуванні:

– Еквівалентне розбиття (Equivalence Partitioning): вхідні дані поділено на класи еквівалентності. Наприклад, для поля "ціна" виокремлено класи: від'ємне значення, нуль, додатне значення.

– Аналіз граничних значень (Boundary Value Analysis): для рейтингу відгуків перевірялись значення 1, 5 та граничні недопустимі 0 і 6.

– Тестування переходів стану: послідовності зміни статусу замовлення ("Нове" → "В обробці" → "Відправлено" → "Виконано") перевірялись на коректність.

– Негативне тестування: перевірялась поведінка системи при введенні некоректних даних, спробі несанкціонованого доступу та помилці платежу.

4.2 Функціональне тестування

Результати функціонального тестування основних модулів системи наведено у таблиці 4.1.

Таблиця 4.1 – Результати функціонального тестування

TC-ID	Тест-кейс	Вхідні дані	Очікуваний результат	Статус
TC-01	Перегляд каталогу без фільтрів	GET /catalog	Відображення 6 товарів, пагінація	PASS
TC-02	Пошук за назвою	q=iPhone	Відображення товарів, що містять "iPhone"	PASS
TC-03	Пошук без результатів	q=zzzzz	Порожній список, повідомлення "нічого не знайдено"	PASS
TC-04	Додавання до кошика	GET /add_to_cart/1	cart_count збільшується на 1, Flash "додано"	PASS
TC-05	Видалення з кошика	GET /remove_from_cart/0	Товар прибирається, перерахунок суми	PASS
TC-06	Оформлення з оплатою карткою	payment_method=card, valid token	Order збережено, статус "Оплачено", email	PASS
TC-07	Помилка платежу картою	card_token=fail_test	Flash-помилка, замовлення НЕ створено	PASS
TC-08	Оформлення "накладений платіж"	payment_method=cod	Order збережено, статус "Очікує оплати"	PASS
TC-09	Реєстрація нового користувача	Унікальний username/email	Запис у БД, редирект на /login	PASS
TC-10	Реєстрація з існуючим email	Вже існуючий email	Flash "Користувач вже існує", без запису	PASS

TC-ID	Тест-кейс	Вхідні дані	Очікуваний результат	Статус
TC-11	Авторизація (коректні дані)	admin / admin123	Сесія встановлена, редирект на sudo	PASS
TC-12	Авторизація (неправильний пароль)	admin / wrongpass	Flash "Невірний логін або пароль"	PASS
TC-13	Відновлення пароля	Правильна секретна відповідь	Пароль оновлено, редирект на /login	PASS
TC-14	Додавання до Wishlist	GET /wishlist/add/1 (авторизований)	Запис у Wishlist, Flash "Додано до обраного"	PASS
TC-15	Доступ до /my-account без входу	Неавторизований GET	Редирект на /login, Flash "Увійдіть"	PASS
TC-16	Доступ до /admin без прав	Неадміністраторська сесія	Редирект на /catalog, Flash "Доступ заборонено"	PASS
TC-17	Додавання відгуку	Авторизований POST, rating=4	Відгук збережено, відображається на сторінці	PASS
TC-18	Порожній коментар відгуку	text=""	Flash "Коментар не може бути порожнім"	PASS
TC-19	Зміна статусу замовлення (адмін)	POST status=Відправлено	Order.status оновлено у БД	PASS
TC-20	Очищення кошика	GET /clear_cart	session["cart"] видалено, кошик порожній	PASS

Усі 20 тест-кейсів пройдено зі статусом PASS. Критичних дефектів не виявлено. Протягом тестування виявлено два незначні зауваження, що були усунуті до завершення роботи:

– При спробі оформити замовлення з порожнім кошиком (якщо користувач вручну переходив на /checkout GET-методом) система не обробляла цей випадок. Додано перевірку ids = session.get("cart", []) та редирект на /cart при порожньому кошику.

– Функція відновлення пароля не перевіряла мінімальну довжину нового пароля. Додано перевірку len(new_password) >= 6 на рівні маршруту.

4.3 Аналіз результатів тестування

Результати функціонального тестування підтвердили коректну роботу системи у відповідності до специфікованих вимог. Всі 12 функціональних вимог, визначених у розділі 1.3, перевірено та підтверджено:

Таблиця 4.2 – Відповідність реалізації функціональним вимогам

Вимога	Тест-кейси	Статус
Каталог із пошуком та пагінацією	TC-01, TC-02, TC-03	ВИКОНАНО
Сесійний кошик	TC-04, TC-05, TC-20	ВИКОНАНО
Оформлення замовлення	TC-06, TC-07, TC-08	ВИКОНАНО
Email-підтвердження	TC-06	ВИКОНАНО
Реєстрація та авторизація	TC-09, TC-10, TC-11, TC-12	ВИКОНАНО
Відновлення пароля	TC-13	ВИКОНАНО
Список бажань	TC-14	ВИКОНАНО
Система відгуків	TC-17, TC-18	ВИКОНАНО
Особистий кабінет	TC-15	ВИКОНАНО

Вимога	Тест-кейси	Статус
Контроль доступу	ТС-15, ТС-16	ВИКОНАНО
Адмін-панель (товари)	ТС-19	ВИКОНАНО
Адмін-панель (замовлення)	ТС-19	ВИКОНАНО

Нефункціональні вимоги щодо безпеки підтверджено: паролі зберігаються у хешованому вигляді (PBKDF2-SHA256), SECRET_KEY та SMTP-дані не присутні у коді репозиторію (зберігаються у .env), двоетапна верифікація адміністратора функціонує коректно. Загальна оцінка якості розробленої системи – відповідає поставленим вимогам.

ВИСНОВКИ

На основі проведеного аналізу ринку рішень електронної комерції та порівняння п'яти основних платформ (WooCommerce, OpenCart, PrestaShop, Shopify, Хорошоп) розроблено інформаційну систему підтримки електронної комерції для малого бізнесу на базі Python/Flask, яка забезпечує повний цикл онлайн-торгівлі та дозволяє малому бізнесу мати повний контроль над функціональністю магазину без залежності від комерційних платформ.

У ході виконання кваліфікаційної роботи отримано такі результати:

1. Проведено аналіз предметної області та сформульовано 12 функціональних і 4 нефункціональні вимоги до системи, що охоплюють всі аспекти роботи інтернет-магазину: каталог, кошик, оформлення замовлень, автентифікацію, адміністрування.

2. Спроєктовано реляційну базу даних на п'яти сутностях (User, Product, Order, Review, Wishlist) з визначенням усіх зв'язків та обмежень цілісності. Розроблено архітектуру за шаблоном MVC з виокремленням сервісного шару для платіжного шлюзу та email-сповіщень.

3. Реалізовано програмне забезпечення з використанням Flask 3.0.0 [1], SQLAlchemy 3.1.1 [2], Flask-Mail та Werkzeug [8]. Система включає: каталог із фільтрацією та пагінацією, сесійний кошик, модуль автентифікації з хешуванням паролів PBKDF2-SHA256, двоетапну верифікацію адміністратора, stub платіжного шлюзу з підтримкою карткового платежу та накладеного платежу, HTML email-підтвердження замовлень.

4. Проведено функціональне тестування за 20 тест-кейсами, що охоплюють усі варіанти використання системи. Всі 20 тест-кейсів пройдено зі статусом PASS. Виявлено та усунено 2 незначні дефекти.

5. Практична цінність розробки полягає у тому, що система може бути безпосередньо розгорнута на VPS-сервері або хмарній платформі (Heroku, Railway, Render) та використана малим бізнесом як основа для повноцінного

інтернет-магазину. Вартість щомісячного хостингу – від $5 до $20 проти $30–300 для SaaS-рішень, що дає економію від $300 до $3000 на рік.

Напрями подальшого розвитку системи:

– міграція з SQLite на PostgreSQL для забезпечення конкурентного доступу та масштабованості;

– реалізація реальної інтеграції з платіжними сервісами LiqPay або WayForPay;

– розробка REST API для мобільного застосунку;

– впровадження системи управління зображеннями товарів через хмарне сховище (AWS S3, Cloudinary);

– додавання модуля аналітики продажів із візуалізацією даних на адмін-панелі;

– впровадження CI/CD-пайплайну для автоматизованого тестування та деплою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd Edition. – O'Reilly Media, 2018. – 316 p.
2. Copeland R. Essential SQLAlchemy. 2nd Edition. – O'Reilly Media, 2016. – 322 p.
3. Lutz M. Programming Python: Powerful Object-Oriented Programming. 4th Edition. – O'Reilly Media, 2010. – 1552 p.
4. Mozilla Developer Network. HTTP – MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP.
5. Bootstrap 5 Documentation [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/
6. Flask Documentation 3.0.x [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/3.0.x/
7. SQLAlchemy ORM Documentation [Електронний ресурс]. – Режим доступу: https://docs.sqlalchemy.org/en/20/orm/
8. Werkzeug Security Documentation [Електронний ресурс]. – Режим доступу: https://werkzeug.palletsprojects.com/en/3.0.x/utils/#werkzeug.security.generate_password_hash
9. Jinja2 Template Documentation [Електронний ресурс]. – Режим доступу: https://jinja.palletsprojects.com/en/3.1.x/
10. LiqPay API Documentation [Електронний ресурс]. – Режим доступу: https://www.liqpay.ua/documentation/api/home
11. WayForPay API Documentation [Електронний ресурс]. – Режим доступу: https://wayforpay.com/uk/documentation
12. OWASP Web Application Security Testing Guide [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/
13. Shopify Partners. Ecommerce Statistics [Електронний ресурс]. – Режим доступу: https://www.shopify.com/enterprise/ecommerce-statistics

14. BuiltWith Technology Trends. E-Commerce Usage Distribution in Ukraine [Електронний ресурс]. – Режим доступу: https://trends.builtwith.com/shop
15. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley Professional, 2002. – 533 p.
16. Freeman E., Robson E. Head First Design Patterns. 2nd Edition. – O'Reilly Media, 2020. – 672 p.
17. Thomas D., Hunt A. The Pragmatic Programmer: Your Journey to Mastery. 20th Anniversary Edition. – Pragmatic Bookshelf, 2019. – 352 p.
18. Держстат України. Статистика оптової та роздрібної торгівлі [Електронний ресурс]. – Режим доступу: https://www.ukrstat.gov.ua/

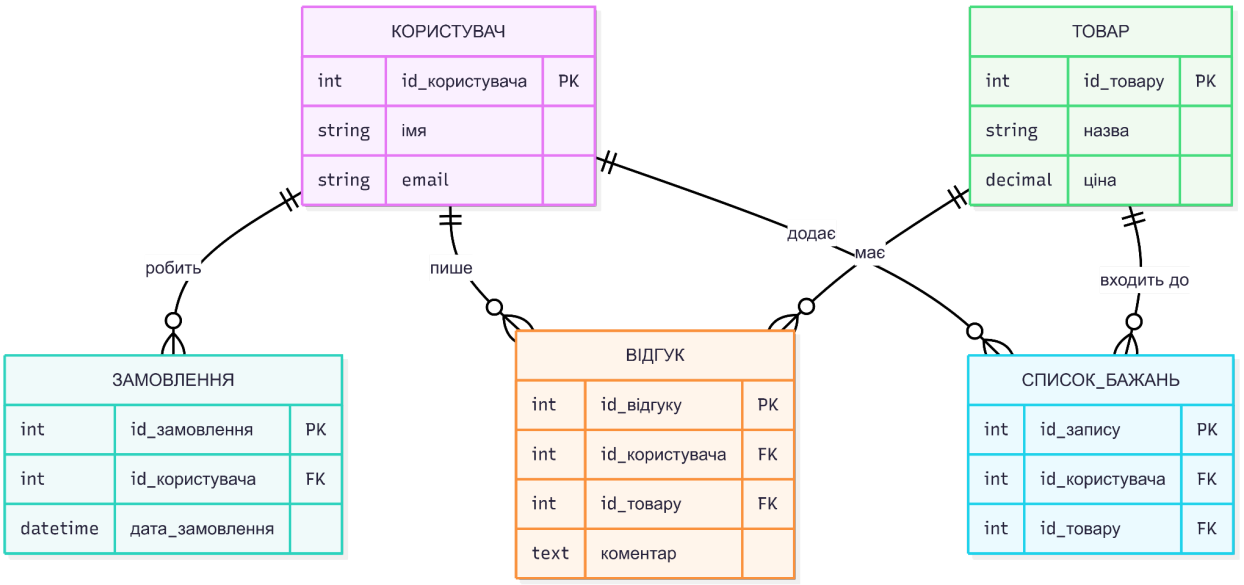
Додаток А

Структура бази даних (ER-діаграма)

ER-діаграма відображає п'ять сутностей системи та їх зв'язки. Нижче наведено текстовий опис усіх таблиць і відносин у нотації "Crow's Foot".

Зв'язок	Тип	Опис
User → Order	1 : N	Один користувач може мати багато замовлень
User → Review	1 : N	Один користувач може залишити багато відгуків
User → Wishlist	1 : N	Один користувач може мати багато позицій у Wishlist
Product → Review	1 : N CASCADE	Видалення товару каскадно видаляє всі відгуки
Product → Wishlist	1 : N	Один товар може бути в Wishlist кількох користувачів

ER-діаграма



Додаток Б

Вихідний код основних модулів

Б.1 Модель Order (замовлення)

```
class Order(db.Model):
    id             = db.Column(db.Integer,
primary_key=True)
    user_id        = db.Column(db.Integer,
db.ForeignKey("user.id"), nullable=True)
    user_name      = db.Column(db.String(100),
nullable=False)
    user_email     = db.Column(db.String(100),
nullable=False)
    user_address   = db.Column(db.String(255),
nullable=False)
    total_amount   = db.Column(db.Float,
nullable=False)
    items          = db.Column(db.Text,
nullable=False)  # JSON
    status         = db.Column(db.String(50),
default="Нове")
    order_date     = db.Column(db.DateTime,
default=datetime.utcnow)
    payment_method = db.Column(db.String(50),
default="Накладений платіж")
    payment_status = db.Column(db.String(50),
default="Очікує оплати")
```

Б.2 Функція ініціалізації бази даних

```
def seed_database():
    db.create_all()
    if not User.query.filter_by(username="admin").first():
        db.session.add(User(
            username="admin", email="admin@techstore.com",

password_hash=generate_password_hash("admin123"),
            is_admin=True,
            secret_question="Місто народження",
            secret_answer=generate_password_hash("кривий
рiг")
        ))
    if Product.query.count() == 0:
        db.session.add_all([...])  # 6 демо-товарів
    db.session.commit()
```

Б.3 Декоратор admin_required

```
def admin_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if "user_id" not in session or not
session.get("is_admin"):
            flash("Доступ заборонено!", "danger")
            return redirect(url_for("catalog"))
        if not session.get("admin_password_confirmed"):
            flash("Підтвердіть пароль адміністратора.",
"warning")
            return redirect(url_for("admin_sudo_auth"))
        return f(*args, **kwargs)
    return decorated
```

Додаток В

Знімки екранних форм

В.1 Опис екранних форм системи TechStore

Сторінка	URL	Призначення	Доступ
Каталог товарів	/catalog	Перегляд та пошук товарів	Публічний
Деталі товару	/product/<id>	Детальна сторінка + відгуки	Публічний
Кошик	/cart	Перелік обраних товарів, підсумок	Публічний
Успішне замовлення	POST /checkout	Підтвердження після оформлення	Публічний
Реєстрація	/register	Форма створення акаунту	Публічний
Авторизація	/login	Форма входу до системи	Публічний
Відновлення пароля	/forgot-password	Форма секретного запитання	Публічний
Особистий кабінет	/my-account	Замовлення та Wishlist	Авторизований
Адмін-панель	/admin	Товари та замовлення	Адміністратор
Адмін-верифікація	/admin/verification	Sudo-підтвердження пароля	Адміністратор
Інформаційні сторінки	/info	Доставка, оплата, контакти	Публічний

Каталог товарів (*див. рис. В.1*)

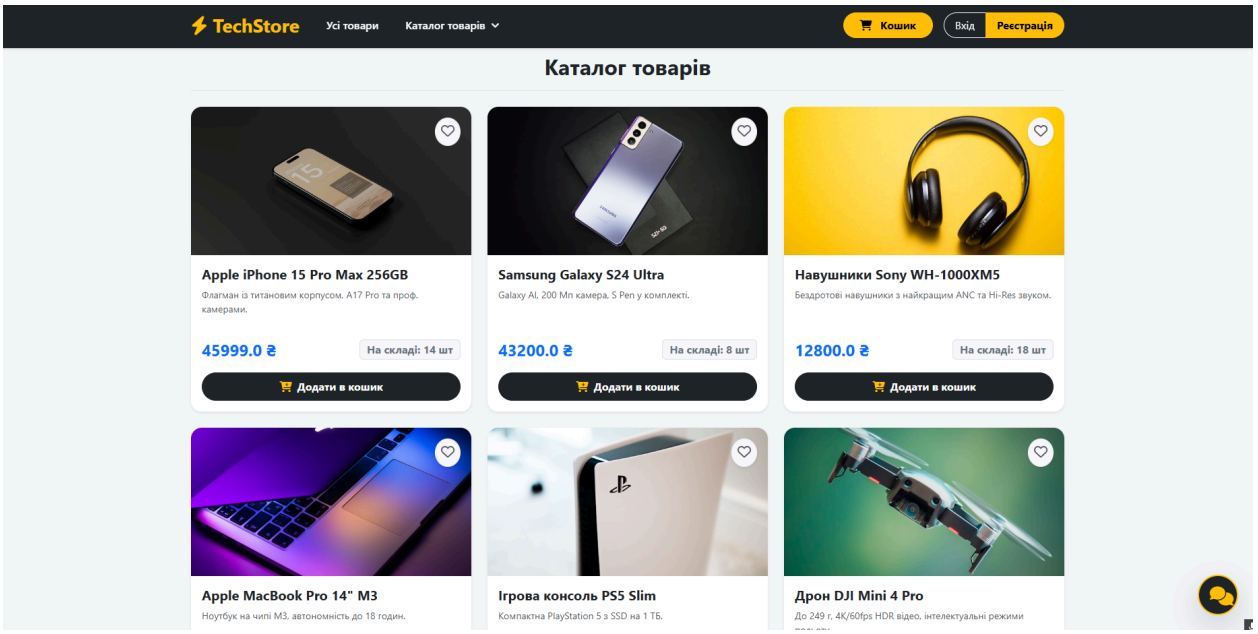


Рисунок В.1 – Каталог товарів

Деталі товару (*див рис. В.2*)

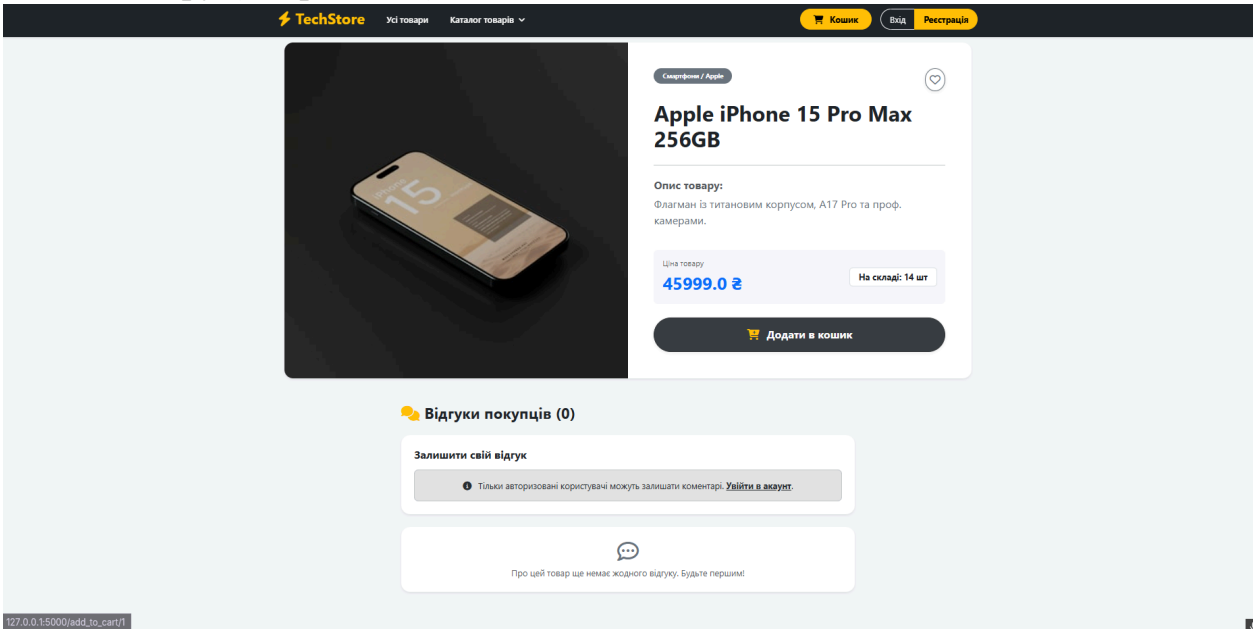


Рисунок В.2 – Деталі товару

Кошик (*див. рис. В.3*)

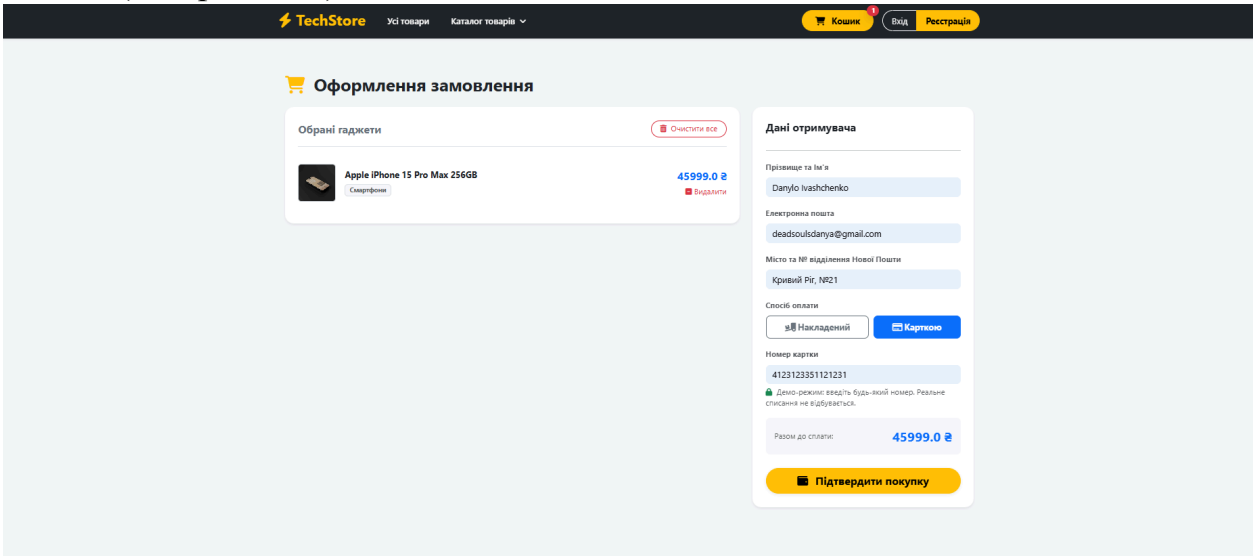


Рисунок В.3 – Кошик

Успішне замовлення (див. рис. В.4)

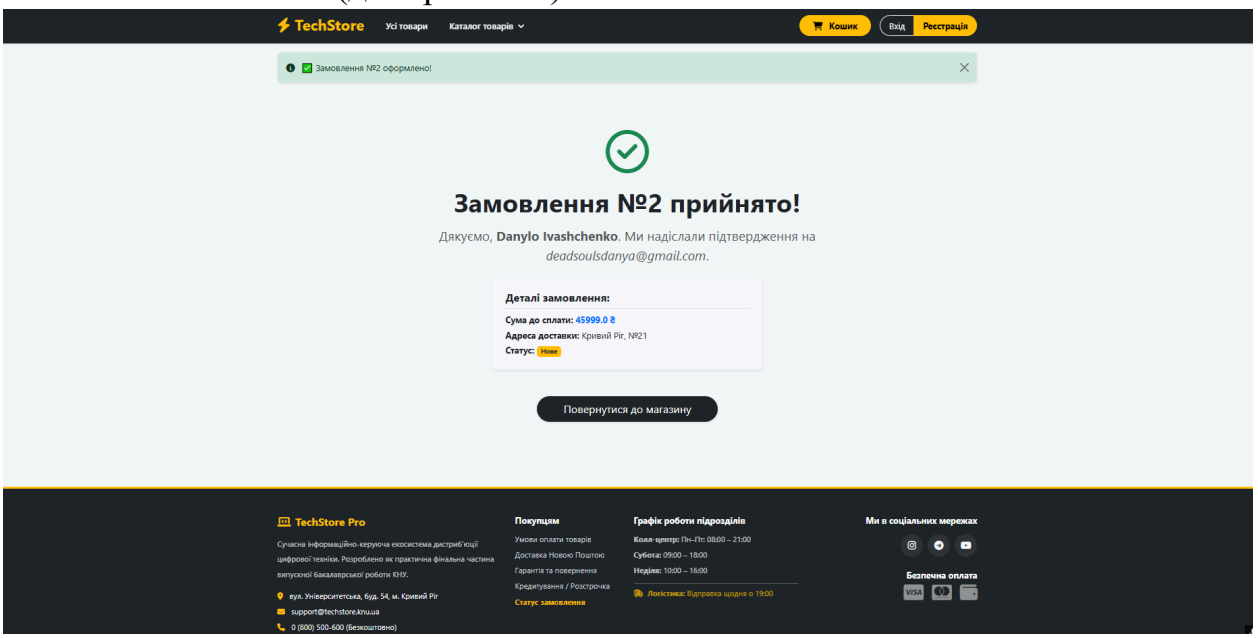


Рисунок В.4 – Успішне замовлення

Реєстрація (див. рис. В.5)

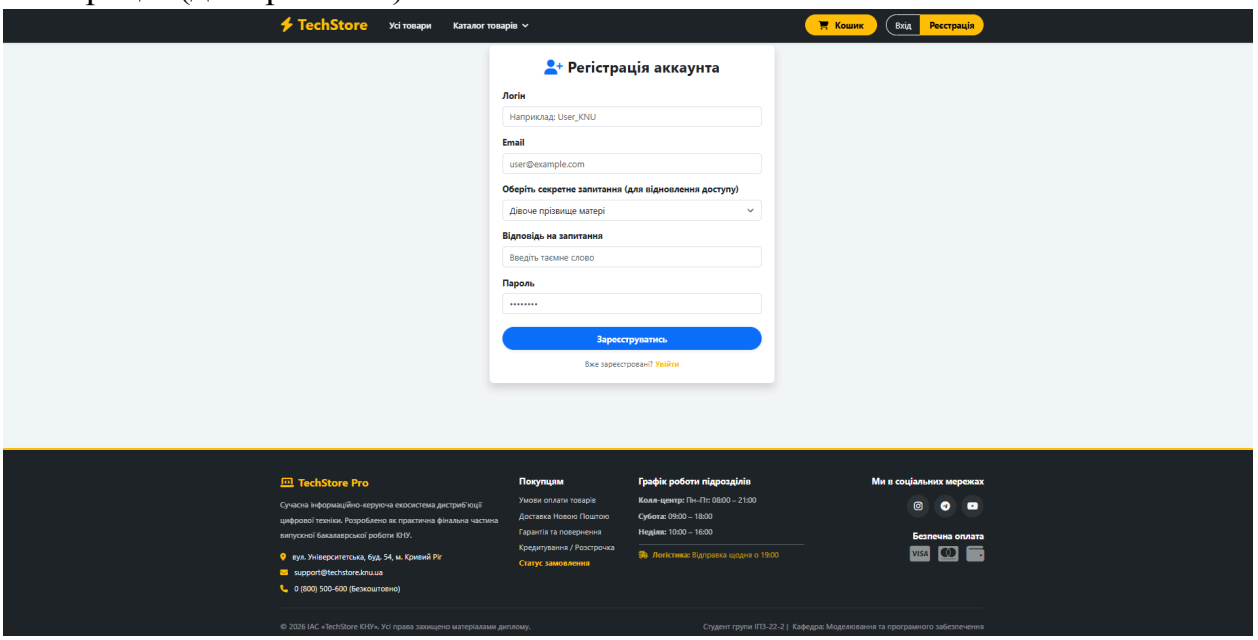


Рисунок В.5 – Реєстрація

Авторизація (див. рис. В.6)

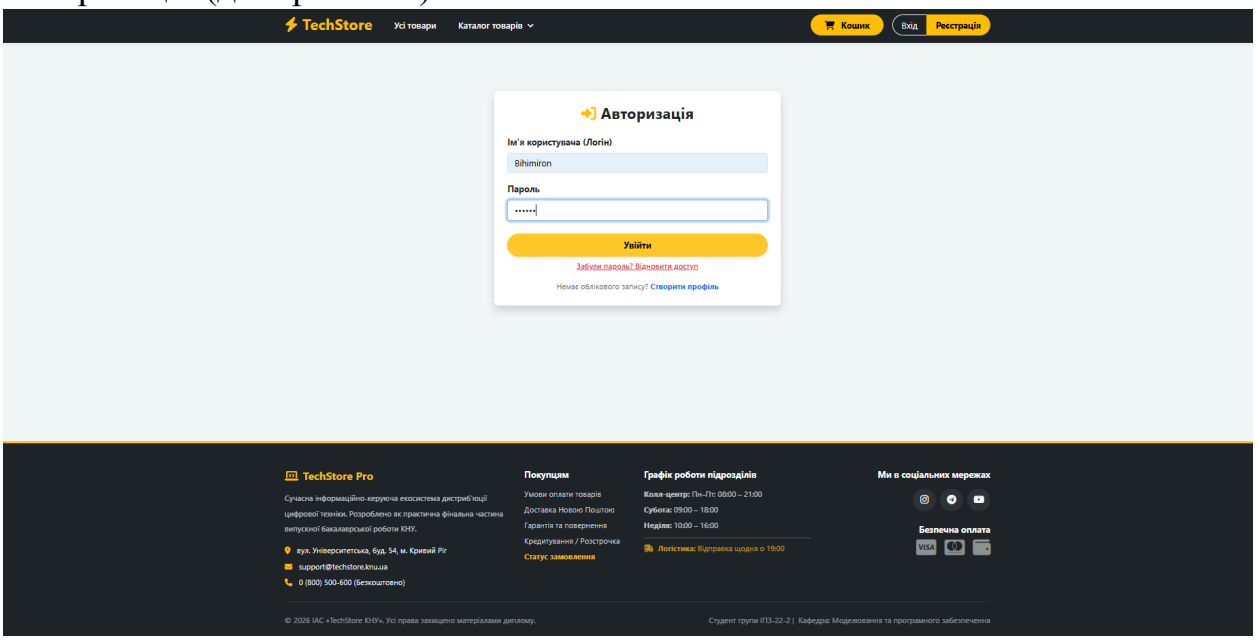


Рисунок В.6 – Авторизація

Відновлення пароля (див. рис. В.7)

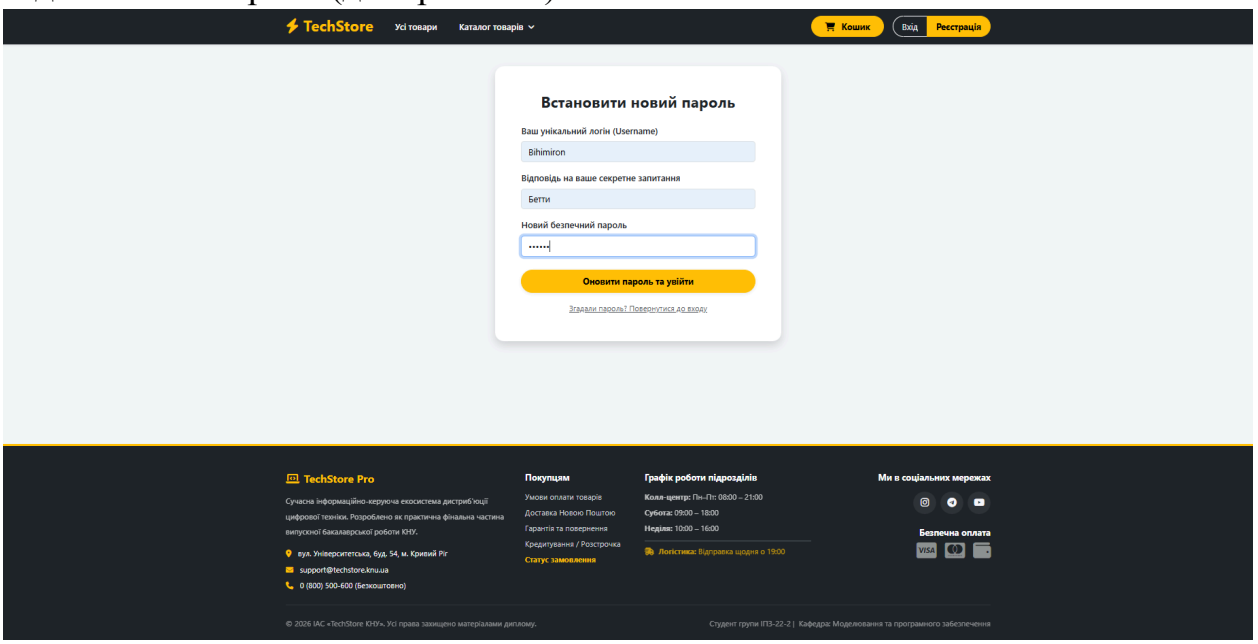


Рисунок В.7 – Відновлення пароля

Особистий кабінет (див. рис. В.8)

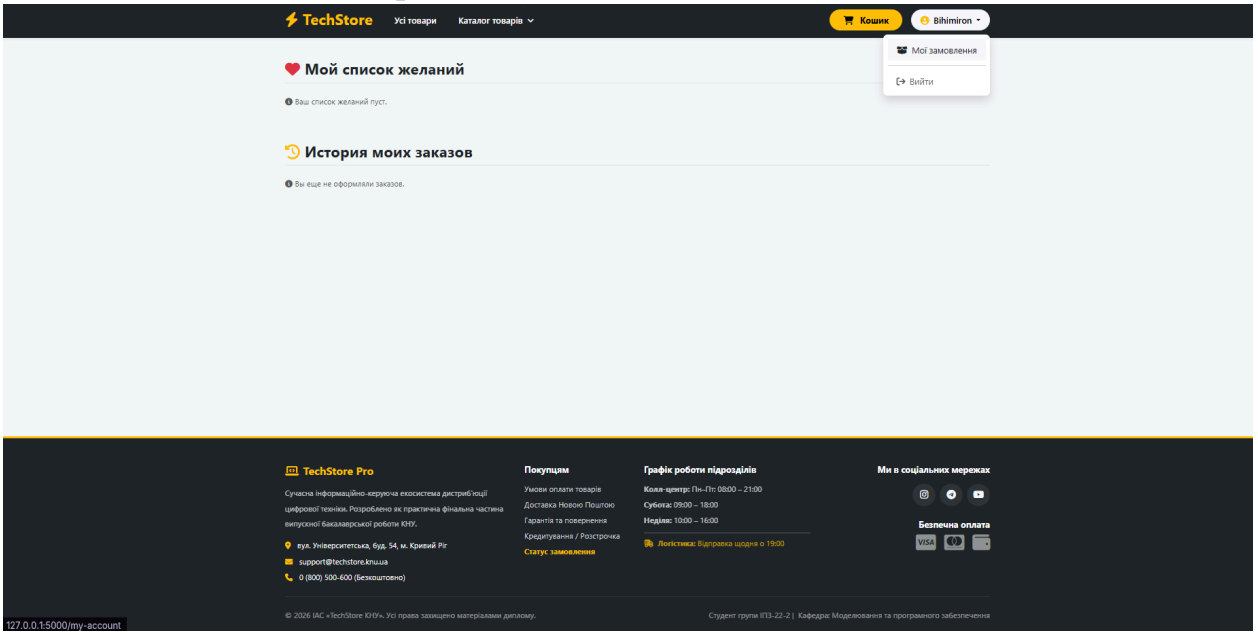


Рисунок В.8 – Особистий кабінет

Адмін-панель (див. рис. В.9)

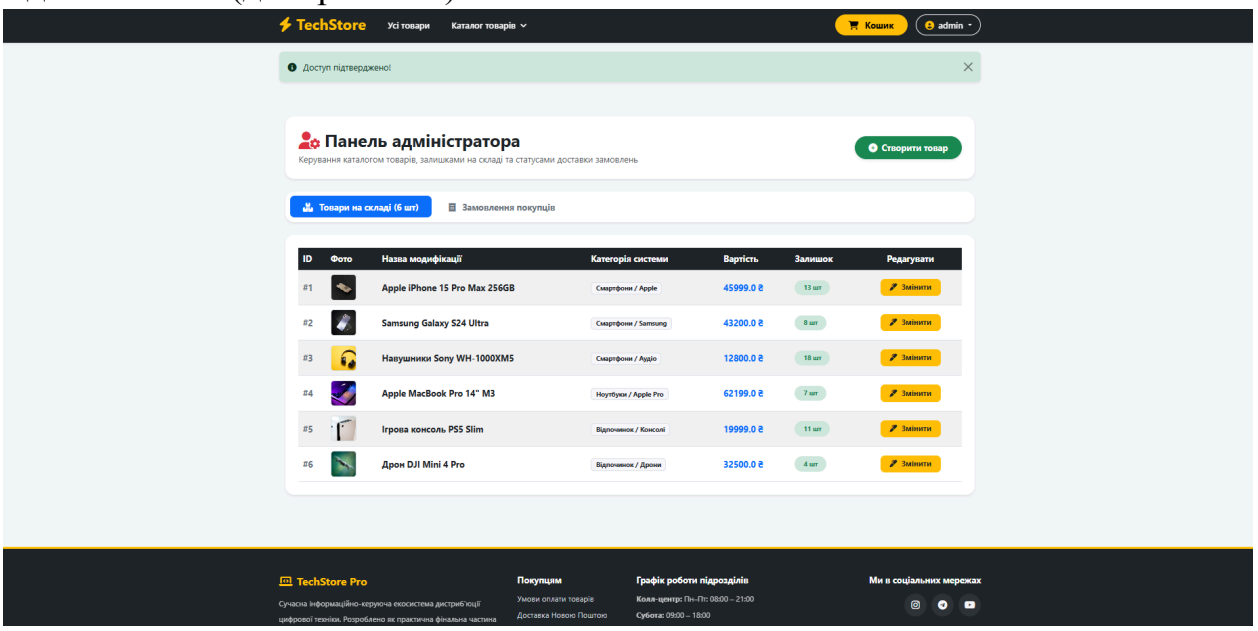


Рисунок В.9 – Адмін-панель

Адмін-верифікація (див. рис. В.10, В.11)

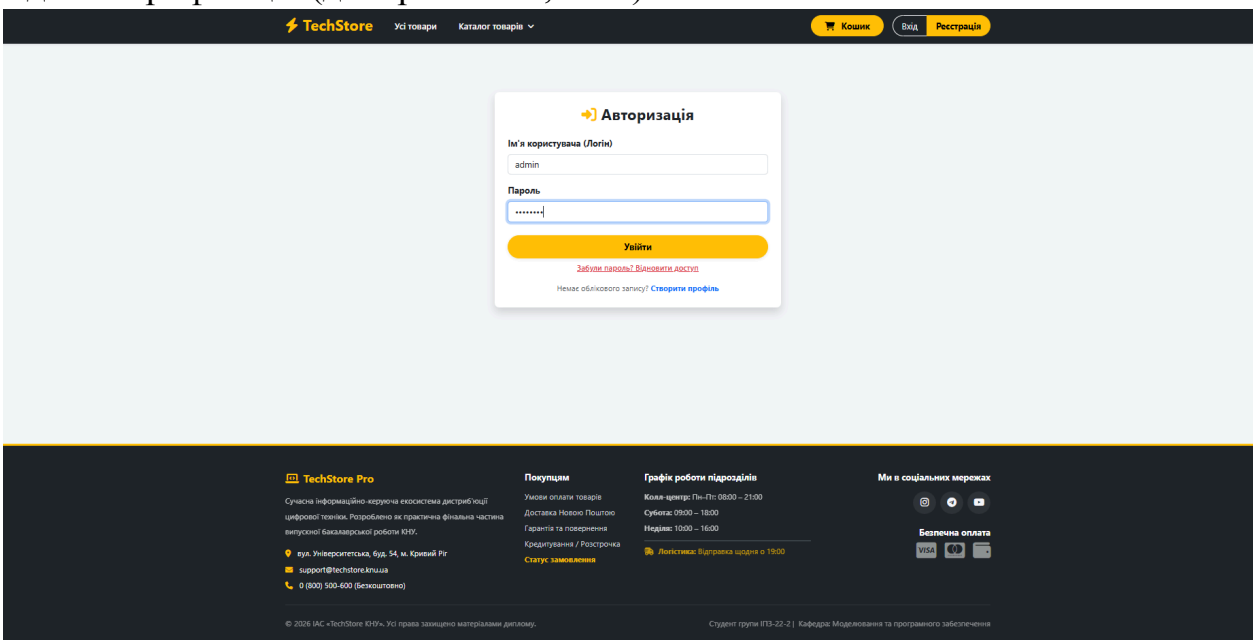


Рисунок В.10 – Адмін-верифікація (крок 1)

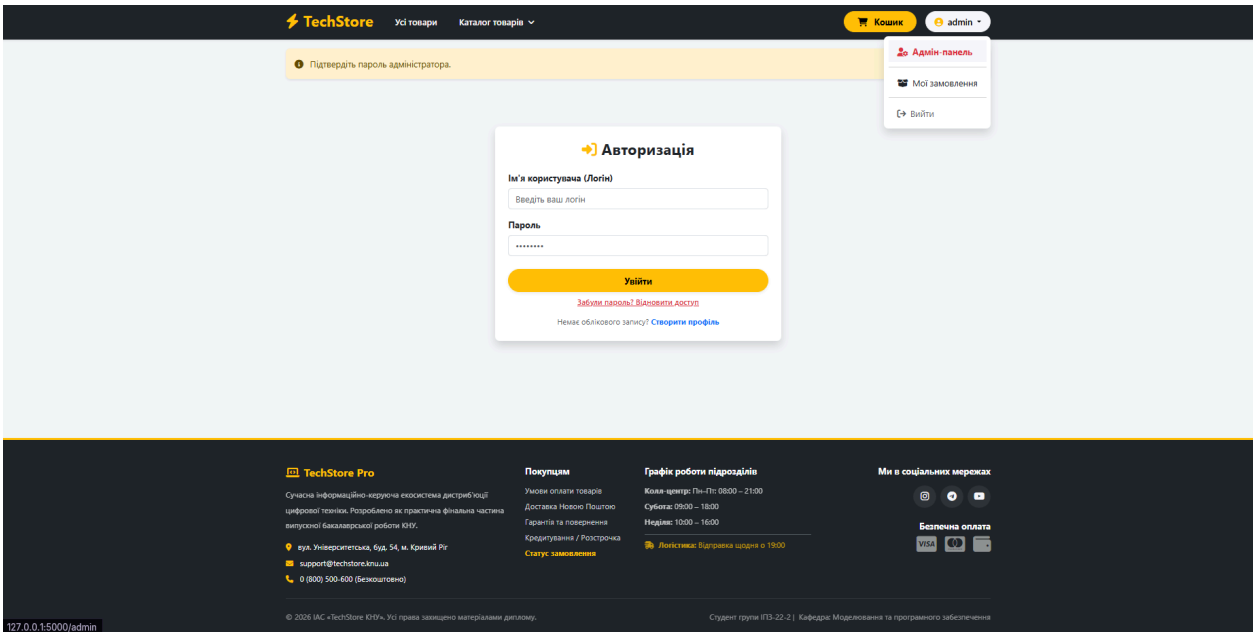


Рисунок В.11 – Адмін-верифікація (крок 2)

Інформаційні сторінки (див. рис. В.12–В.16)

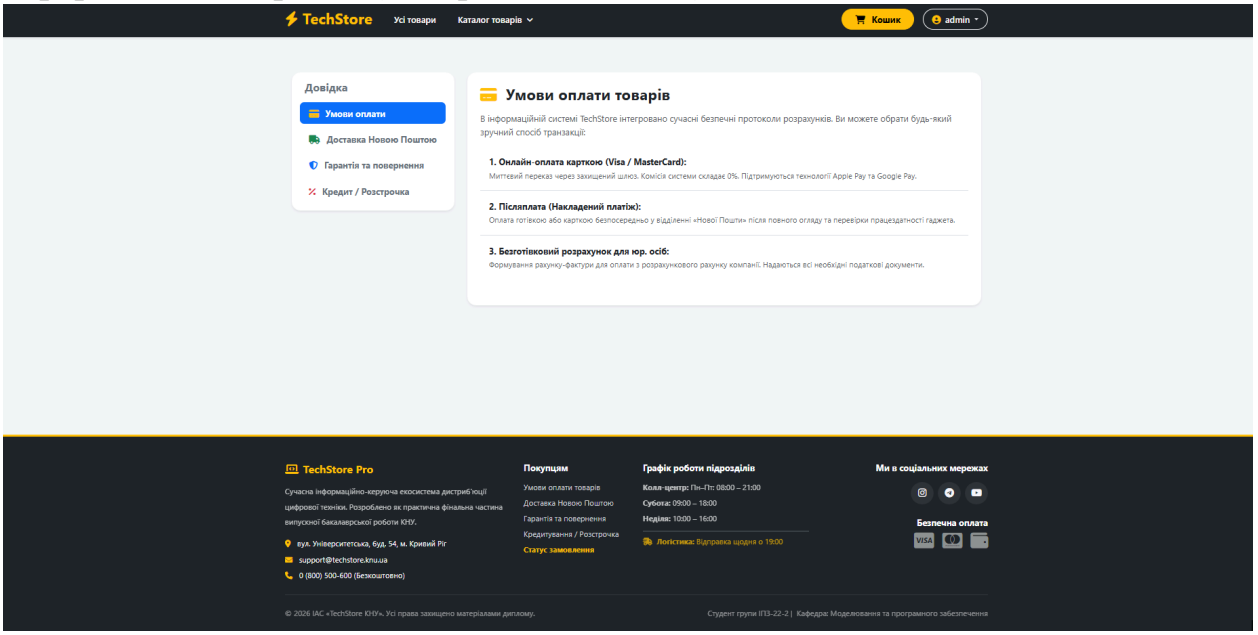


Рисунок В.12 – Інформаційна сторінка (1)

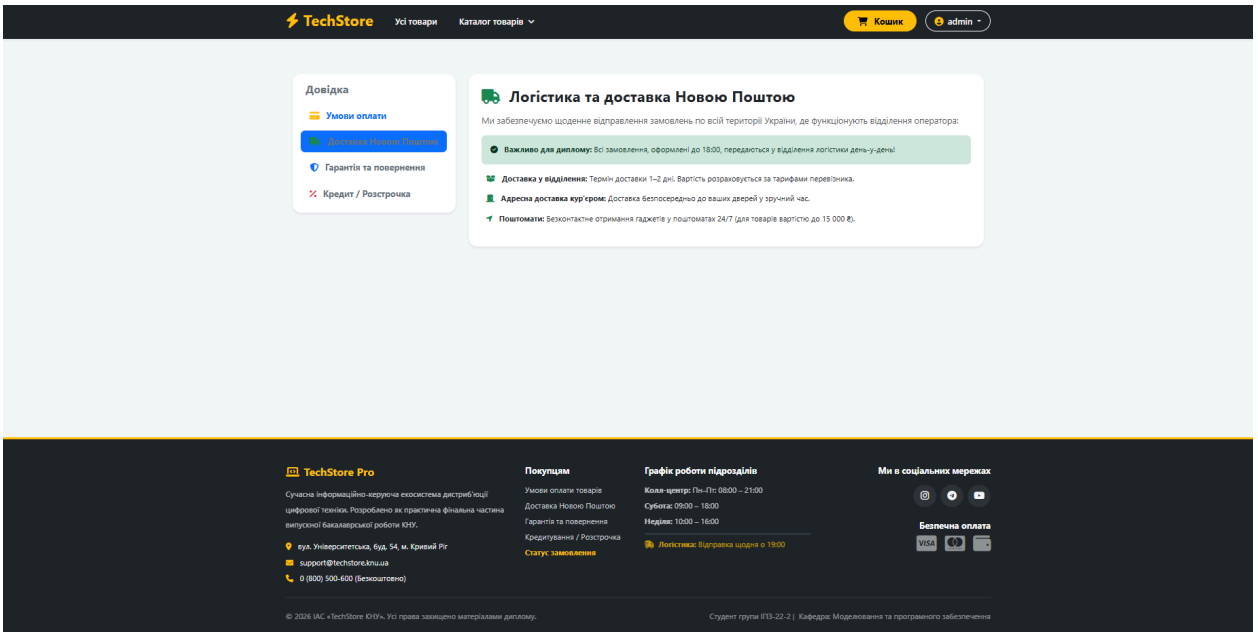


Рисунок В.13 – Інформаційна сторінка (2)

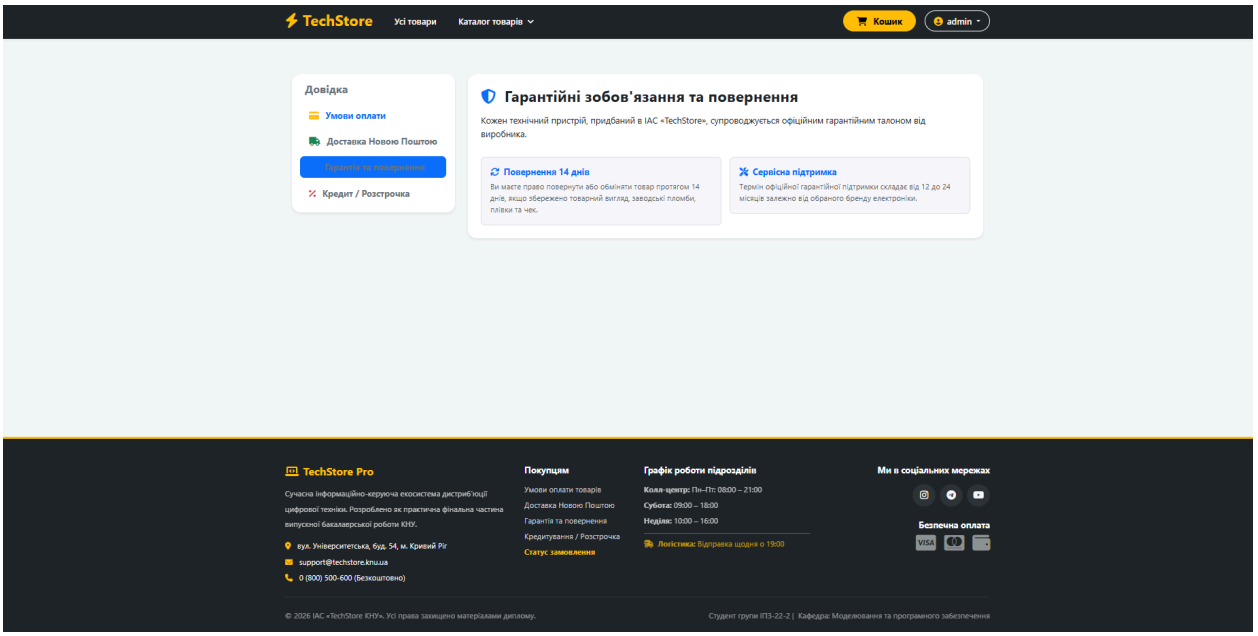


Рисунок В.14 – Інформаційна сторінка (3)

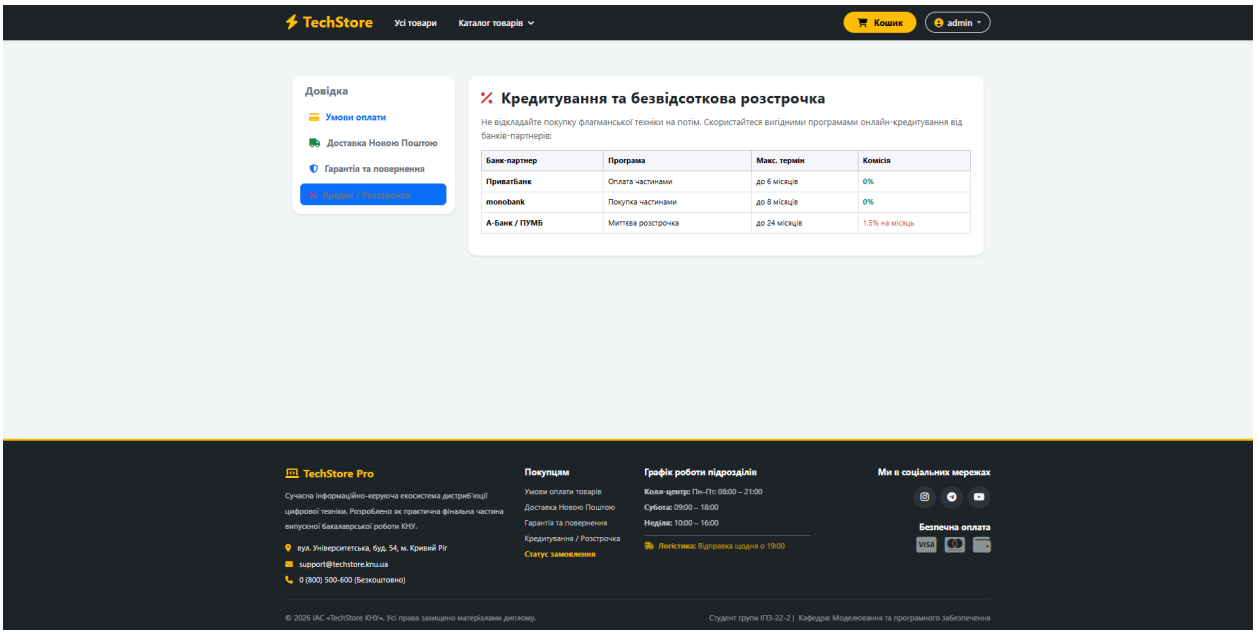


Рисунок В.15 – Інформаційна сторінка (4)

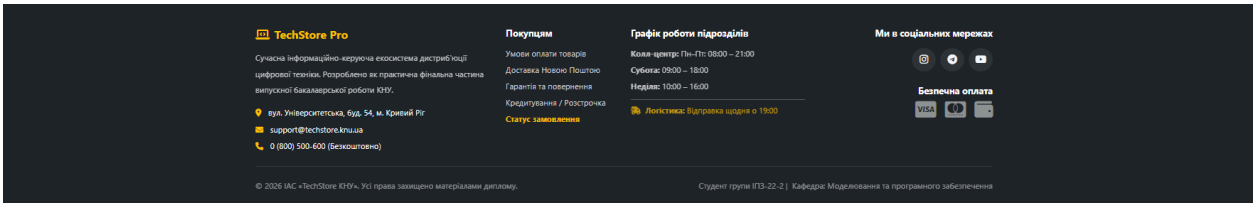


Рисунок В.16 – Інформаційна сторінка (5)