

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

ІНТЕРНЕТ-МАРКЕТПЛЕЙС НА ОСНОВІ PYTHON

ONLINE MARKETPLACE BASED ON PYTHON

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КН-42
Кузьмич Олег Ростиславович

(підпис)

д.пед.н., професор
Тулашвілі Юрій Йосипович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2026 р.
Гарант освітньої програми:
д.пед.н., професор
Тулашвілі Юрій Йосипович

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*

Кафедра *комп'ютерних наук*

Ступінь вищої освіти: *бакалавр*

Галузь знань: *12 Інформаційні технології*

Спеціальність: *122 Комп'ютерні науки*

Освітня програма: *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ

Завідувач кафедри

Валерій ЛІЩИНА

«16» січня 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ПЕРШОГО (БАКАЛАВРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Кузьмич Олег Ростиславович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Інтернет-маркетплейс на основі Python»

Керівник д.пед.н., професор Тулашвілі Юрій Йосипович

затверджені наказом закладу вищої освіти від «16» січня 2026 р. № 32/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи «02» червня 2026 р.

3. Вихідні дані до роботи: *предметна область електронної комерції; вимоги до інтернет-маркетплейсу; аналіз аналогічних програмних рішень; документація Python, Flask, React, SQLite, SQLAlchemy та AWS SageMaker; наукові джерела з теми електронної комерції, маркетплейсів і вебзастосунків.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та надійнісних параметрів проекрованої системи, функціонально-структурна схема роботи об'єкта проектування.

5. Перелік графічного матеріалу: 21 рисунок, 13 лістингів, 3 таблиці.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
<i>Аналіз проблеми за темою роботи та постановка завдань дослідження</i>	<i>Тулашвілі Ю. Й.</i>		
<i>Теоретичне дослідження</i>	<i>Тулашвілі Ю. Й.</i>		
<i>Практична реалізація</i>	<i>Тулашвілі Ю. Й.</i>		
<i>Спеціальні розрахунки</i>	<i>Тулашвілі Ю. Й.</i>		
<i>Показник запозичень тексту</i>	%		
<i>Інструментальна перевірка</i>	<i>Кошеляк В. А.</i>		
<i>Нормоконтроль</i>	<i>Сачук В. О.</i>		
<i>Гарант ОПП</i>	<i>Тулашвілі Ю. Й.</i>		

7. Дата видачі завдання «16» січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	<i>Провести огляд літературних джерел по темі кваліфікаційної роботи</i>	<i>до 17.02.2026 р</i>	
2	<i>Провести аналіз загальної проблеми і вибір напрямків дослідження</i>	<i>до 28.02.2025 р.</i>	
3	<i>Розробити функціональну схему роботи програмного продукту</i>	<i>до 12.03.2026 р</i>	
4	<i>Описати засоби розробки об'єкта проектування</i>	<i>до 26.03.2026 р.</i>	
5	<i>Практична реалізація об'єкта проектування</i>	<i>до 30.04.2026 р.</i>	
6	<i>Провести спеціальні розрахунки</i>	<i>до 18.05.2026 р.</i>	
7	<i>Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі</i>	<i>до 02.06.2026 р.</i>	

Здобувач вищої освіти _____ Олег КУЗЬМИЧ

Керівник роботи _____ Юрій ТУЛАШВІЛІ

АНОТАЦІЯ

Кузьмич О. Р. Інтернет-маркетплейс на основі Python. Рукопис. Кваліфікаційна робота бакалавра ОП «Комп'ютерні науки». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі виконано аналіз предметної області електронної комерції, розглянуто особливості функціонування інтернет-маркетплейсів, наведено приклади аналогічних рішень та обґрунтовано вибір напрямку розробки. Другий розділ містить специфікацію вимог до розроблюваної системи, опис функціонально-структурної схеми роботи об'єкта проектування, сценарії використання, UML-діаграми, блок-схеми алгоритмів та модель бази даних. У третьому розділі здійснено практичну реалізацію інтернет-маркетплейсу, включаючи розробку клієнтської частини, серверної логіки, API, механізму роботи з товарами, кошиком, замовленнями та відгуками. Окрему увагу приділено інтеграції AI-модуля для аналізу тональності відгуків користувачів. Четвертий розділ містить спеціальні розрахунки та обґрунтування заходів захисту інформації, зокрема питання автентифікації користувачів, розмежування доступу за ролями, безпечної обробки даних та захисту взаємодії між клієнтською і серверною частинами.

У висновках узагальнено результати виконаної роботи та визначено перспективи подальшого розвитку системи.

Ключові слова: інтернет-маркетплейс, електронна комерція, вебзастосунок, React, Flask, Python, SQLite, API, AI-модель, аналіз відгуків, автентифікація, авторизація.

ANNOTATION

Oleh Kuzmych. Online Marketplace Based on Python. Manuscript. Bachelor's qualification paper in the study program «Computer Science» Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification paper consists of an introduction, four chapters, conclusions, a list of references and appendices.

The first chapter analyzes the subject area of e-commerce, considers the main features of online marketplaces, reviews similar software solutions and substantiates the chosen development direction. The second chapter contains the specification of requirements for the developed system, a description of the functional and structural scheme of the designed object, use case scenarios, UML diagrams, algorithm flowcharts and the database model. The third chapter presents the practical implementation of the online marketplace, including the development of the client side, server logic, API, product management, cart, order processing and review functionality. Special attention is paid to the integration of an AI module for sentiment analysis of user reviews. The fourth chapter contains special calculations and justification of information protection measures, including user authentication, role-based access control, secure data processing and protection of communication between the client and server parts. The conclusions summarize the results of the work and define the prospects for further development of the system.

Keywords: online marketplace, e-commerce, web application, React, Flask, Python, SQLite, API, AI model, review analysis, authentication, authorization.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Аналіз сучасного стану проблеми	10
1.2 Аналіз аналогічних програм, баз даних, систем, обґрунтування вибраного напрямку та вибір методів рішення	12
1.3 Аналіз і вибір засобів проєктування інтернет-маркетплейсу	14
1.4 Постановка завдання на кваліфікаційну роботу бакалавра	16
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	18
2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	18
2.2 Функціонально-структурна схема роботи об'єкта проєктування	19
2.3 Створення моделі бази даних	30
РОЗДІЛ 3 РОЗРОБКА ІНТЕРНЕТ-МАРКЕТПЛЕЙСУ	36
3.1 Практична реалізація об'єкта проєктування. Побудова блок-схем модулів програми	36
3.2 Тестування та налагодження системи інтернет-маркетплейсу	50
3.3 Посібник з інсталяції та використання програмного продукту	55
РОЗДІЛ 4 СПЕЦІАЛЬНІ РОЗРАХУНКИ	58
4.1 Захист інформації в інтернет-маркетплейсі	58
ВИСНОВКИ	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64
ДОДАТКИ	66

ВСТУП

Сучасний розвиток інформаційних технологій суттєво змінив підходи до організації торгівлі, взаємодії між продавцями та покупцями, а також способи представлення товарів у цифровому середовищі. Значна частина торговельних процесів переходить в онлайн-формат, що зумовлює потребу у створенні зручних, швидких і функціональних веб-застосунків для електронної комерції. Одним із найбільш поширених типів таких систем є інтернет-маркетплейс, який забезпечує взаємодію різних груп користувачів у межах єдиної програмної платформи.

Інтернет-маркетплейс відрізняється від звичайного інтернет-магазину тим, що передбачає розмежування функцій між різними ролями користувачів. Гість може переглядати відкриту частину сайту, каталог і сторінки товарів. Покупець після авторизації отримує можливість додавати товари до кошика, оформлювати замовлення, переглядати їх стан і залишати відгуки після підтвердженої покупки. Продавець, у свою чергу, працює із seller-панеллю, у якій може керувати власними товарами та замовленнями, що стосуються його товарних позицій.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю розробки програмних рішень, які поєднують зручний користувацький інтерфейс, клієнт-серверну архітектуру, роботу з базою даних, рольову модель доступу та можливість подальшого розширення функціональності. Для інтернет-маркетплейсу важливо реалізувати не лише відображення товарів, а й основні бізнес-процеси електронної комерції: перегляд каталогу, роботу з кошиком, оформлення замовлення, обробку seller-функцій, створення відгуків і контроль доступу до окремих дій залежно від ролі користувача.

У межах розроблюваної системи реалізовано AI-модуль аналізу тональності відгуків. Після створення відгуку текст коментаря передається до модуля аналізу, який визначає його позитивний або негативний характер. Результат класифікації зберігається разом із відгуком і може використовуватися

для подальшої оцінки реакції покупців на товари. Це демонструє практичне використання засобів штучного інтелекту у сфері електронної комерції.

Метою кваліфікаційної роботи бакалавра є розробка інтернет-маркетплейсу на основі Python, який забезпечує основні сценарії взаємодії гостя, покупця та продавця, підтримує роботу з товарами, кошиком, замовленнями, відгуками та передбачає інтеграцію AI-модуля для аналізу тональності текстових коментарів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область електронної комерції та особливості функціонування інтернет-маркетплейсів;
- дослідити аналогічні програмні рішення та обґрунтувати вибір напряму розробки власної системи;
- обґрунтувати вибір технологій, методів і засобів реалізації frontend-, backend-частини, бази даних та AI-модуля;
- сформулювати функціональні й нефункціональні вимоги до інтернет-маркетплейсу;
- розробити функціонально-структурну схему, UML-діаграми, блок-схеми алгоритмів і модель бази даних;
- реалізувати клієнтську та серверну частини вебзастосунку з підтримкою ролей гостя, покупця і продавця;
- реалізувати основні функції маркетплейсу: каталог товарів, кошик, оформлення замовлень, seller-панель, відгуки та AI-аналіз тональності;
- провести тестування основних сценаріїв роботи системи, AI-модуля та розглянути питання захисту інформації.

Об'єктом дослідження є веборієнтовані інформаційні системи для електронної комерції, що забезпечують взаємодію між покупцями та продавцями.

Предметом дослідження є інтернет-маркетплейс та методи і засоби для його створення з клієнт-серверною архітектурою з рольовою моделлю доступу, системою замовлень, відгуками та AI-аналізом текстових коментарів.

Новизна роботи полягає у поєднанні базової логіки інтернет-маркетплейсу з AI-модулем автоматичного аналізу тональності відгуків покупців. У розробленій системі відгук не лише зберігається у базі даних, а й додатково обробляється AI-модулем, який визначає його позитивний або негативний характер. Також у системі реалізовано обмеження, за яким покупець може залишити відгук лише після підтвердженої оплати замовлення, що підвищує достовірність користувацьких коментарів.

Практичне значення роботи полягає у створенні функціонального прототипу інтернет-маркетплейсу, який демонструє основні бізнес-процеси електронної комерції на рівні MVP. Розроблений застосунок може бути використаний як основа для подальшого розвитку e-commerce платформи або як навчально-практичний приклад побудови вебзастосунку з frontend-частиною, backend API, базою даних, рольовою моделлю користувачів, системою замовлень, відгуками та AI-модулем аналізу текстових коментарів.

Апробацію результатів кваліфікаційної роботи здійснено шляхом представлення тез на Міжнародній науково-практичній конференції «Сучасні інформаційні технології: від теорії до практики 2026» [1] (додаток А).

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі виконано аналіз предметної області, розглянуто особливості інтернет-маркетплейсів, проаналізовано аналогічні рішення та сформульовано постановку завдання. У другому розділі наведено специфікацію вимог до системи, функціонально-структурні схеми, UML-діаграми, блок-схеми алгоритмів і модель бази даних. У третьому розділі описано практичну реалізацію frontend- і backend-частин, роботу з API, тестування та інструкцію з використання програмного продукту. У четвертому розділі розглянуто спеціальні розрахунки, пов'язані із захистом інформації, автентифікацією, авторизацією, контролем доступу та безпечною взаємодією між клієнтською і серверною частинами.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасного стану проблеми

Електронна комерція є одним із найбільш поширених напрямів застосування сучасних інформаційних технологій. Її розвиток пов'язаний зі зростанням кількості користувачів мережі Інтернет, поширенням мобільних пристроїв, розвитком електронних платежів, хмарних сервісів і цифрових каналів комунікації між бізнесом та споживачами. Онлайн-торгівля дає змогу користувачам здійснювати пошук, вибір і замовлення товарів незалежно від місця перебування та часу доби. Сучасний ринок електронної комерції в Україні продовжує розвиватися, зокрема завдяки поширенню інтернет-магазинів, онлайн-комерції та маркетплейсів [2].

Одним із важливих різновидів систем електронної комерції є інтернет-маркетплейс. На відміну від класичного інтернет-магазину, маркетплейс є платформою, у якій взаємодіють декілька груп користувачів. У межах такої системи доцільно виділяти гостя, покупця та продавця. Гість має доступ до відкритої частини сайту, покупець після авторизації може працювати з кошиком, замовленнями та відгуками, а продавець отримує доступ до окремої seller-панелі для керування товарами й обробки замовлень, що стосуються його товарних позицій.

Сучасний інтернет-маркетплейс повинен забезпечувати повний цикл взаємодії користувача із системою. До такого циклу належать перегляд каталогу, відкриття сторінки товару, додавання товару до кошика, оформлення замовлення, перегляд історії покупок, робота з адресами доставки та створення відгуків. Для продавця важливими є створення і редагування товарів, перегляд замовлень та підтвердження стану оплати. Усі ці дії формують інформаційні потоки між клієнтською частиною, backend API та базою даних, тому система має бути побудована за клієнт-серверним принципом. Під час проєктування каталогу важливо враховувати не лише технічне відображення товарів, а й логіку

формування асортименту, оскільки структура товарних пропозицій впливає на сприйняття платформи користувачем і ефективність онлайн-торгівлі [3].

Важливою вимогою до сучасних вебзастосунків є зручність користувацького інтерфейсу. Інтерфейс маркетплейсу має бути зрозумілим для різних категорій користувачів і підтримувати основні сценарії роботи без зайвого ускладнення. Гість повинен швидко перейти до каталогу та сторінки товару, покупець – виконати дії з кошиком і замовленням, а продавець – отримати доступ до інструментів керування товарами та замовленнями. Якщо структура сторінок є нелогічною або шлях до покупки занадто складний, користувач може залишити сайт без оформлення замовлення.

Окреме значення має адаптивність інтерфейсу. Значна частина користувачів взаємодіє з онлайн-магазинами та маркетплейсами через мобільні пристрої, тому сторінки каталогу, товару, кошика та оформлення замовлення мають коректно відображатися на екранах різного розміру. Для цього клієнтська частина повинна мати гнучку структуру компонентів і навігацію, придатну як для desktop-пристроїв, так і для мобільних екранів.

Ще одним важливим елементом маркетплейсу є система відгуків. Відгуки допомагають покупцям оцінити якість товару та досвід інших користувачів, однак потребують контролю достовірності. Доцільним є підхід, за якого залишити відгук може лише покупець, який фактично оформив замовлення і для якого було підтверджено оплату. Додатково актуальним є автоматизований аналіз текстових коментарів. Використання AI-модуля для визначення тональності відгуку дозволяє класифікувати коментар як позитивний або негативний і надалі використовувати ці дані для оцінки реакції покупців на товари.

Розвиток е-комерції в Україні супроводжується не лише технологічними можливостями, а й низкою проблем, серед яких кібербезпекові ризики, шахрайство, недостатній рівень довіри користувачів до онлайн-платформ та інфраструктурні обмеження [4]. Тому під час розробки інтернет-маркетплейсу

важливо враховувати не тільки функціональність, а й питання безпеки, контролю доступу, достовірності дій користувачів і захисту даних.

Отже, проблема розробки інтернет-маркетплейсу є актуальною, оскільки така система поєднує декілька важливих аспектів програмної інженерії: побудову клієнт-серверної архітектури, роботу з базою даних, реалізацію ролей користувачів, обробку замовлень, створення адаптивного інтерфейсу та інтеграцію інтелектуальних засобів аналізу даних. У межах кваліфікаційної роботи доцільно реалізувати навчально-практичний прототип, який не повторює повністю функціональність великих комерційних платформ, але демонструє основні процеси роботи маркетплейсу: перегляд товарів, оформлення замовлення, роботу seller-панелі, створення відгуків і AI-аналіз їх тональності.

1.2 Аналіз аналогічних програм, баз даних, систем, обґрунтування вибраного напрямку та вибір методів рішення

Для визначення функціональних вимог до розроблюваного програмного продукту було розглянуто типові можливості існуючих систем електронної комерції. До аналогічних рішень можна віднести великі міжнародні маркетплейси, локальні платформи онлайн-торгівлі та сервіси оголошень, зокрема Amazon, Etsy, Rozetka, Prom та OLX. Попри відмінності у масштабі, цільовій аудиторії та бізнес-моделі, такі системи мають спільну основу: каталог товарів або оголошень, сторінку окремої позиції, обліковий запис користувача, механізм взаємодії покупця з продавцем і засоби керування розміщеними товарами.

Типовий маркетплейс містить головну сторінку, каталог товарів, сторінку окремого товару, систему пошуку або фільтрації, кошик чи механізм оформлення замовлення, особистий кабінет користувача, історію замовлень і блок відгуків. Для продавців у таких системах передбачаються окремі інструменти керування товарами: створення нових позицій, редагування опису, ціни, кількості, категорії та перегляд замовлень, що стосуються їхніх товарів.

Перевагою великих маркетплейсів є широкий функціональний набір, масштабованість, підтримка різних способів оплати та доставки, розвинені механізми пошуку, фільтрації й рекомендацій. Водночас такі системи є складними для повної реалізації в межах кваліфікаційної роботи, оскільки потребують розгалуженої серверної інфраструктури, адміністрування великого обсягу даних, інтеграції платіжних сервісів, служби доставки та додаткових аналітичних модулів. Тому для розроблюваного застосунку доцільним є MVP-підхід, за якого реалізуються основні бізнес-процеси маркетплейсу без надмірного ускладнення системи.

Під час аналізу аналогічних систем було визначено, що для розроблюваного застосунку необхідно реалізувати сценарії гостя, покупця та продавця. Для гостя важливими є перегляд головної сторінки, каталогу та сторінки товару без обов'язкової авторизації. Для покупця ключовими є додавання товарів до кошика, оформлення замовлення, перегляд власних замовлень і створення відгуків після підтвердженої покупки. Для продавця важливими є керування товарами, перегляд замовлень, що містять його товари, та підтвердження стану оплати.

Окремо було враховано підхід до створення відгуків. У багатьох системах електронної комерції відгук має залишати не випадковий відвідувач, а користувач, який фактично здійснив покупку. Це зменшує кількість недостовірних або випадкових коментарів. У розроблюваній системі цей принцип реалізовано через перевірку оплаченої покупки: покупець може залишити відгук лише після того, як продавець підтвердив оплату замовлення, а відповідний товар присутній у цьому замовленні.

Для зберігання даних у подібній системі доцільно використовувати реляційну модель, оскільки сутності маркетплейсу мають чітко визначені зв'язки. Користувачі пов'язані з ролями, адресами, кошиком, замовленнями та відгуками. Товари пов'язані з категоріями, продавцями, зображеннями, позиціями кошика, позиціями замовлення та відгуками. Така структура добре

описується за допомогою таблиць, зовнішніх ключів і зв'язків один-до-одного, один-до-багатьох та багато-до-одного.

У результаті аналізу було обрано напрям розробки веборієнтованого інтернет-маркетплейсу з клієнт-серверною архітектурою. Такий підхід дозволяє розділити відповідальність між frontend-частиною, яка відповідає за взаємодію з користувачем, і backend-частиною, яка забезпечує обробку запитів, бізнес-логіку, перевірку ролей, роботу з базою даних та інтеграцію AI-модуля аналізу відгуків. Обраний напрям дає змогу створити навчально-практичний прототип, який відображає основну логіку маркетплейсу та може бути розширений у майбутньому.

1.3 Аналіз і вибір засобів проєктування інтернет-маркетплейсу

Для проєктування та реалізації інтернет-маркетплейсу обрано технології, які дають змогу розділити клієнтську частину, серверну логіку, базу даних і AI-модуль аналізу відгуків. Такий вибір відповідає клієнт-серверній архітектурі системи та дозволяє реалізувати основні сценарії роботи гостя, покупця і продавця без надмірного ускладнення програмної структури.

Клієнтську частину реалізовано з використанням React, який дає змогу будувати інтерфейс на основі компонентного підходу [5]. Для запуску середовища розробки та збірки frontend-частини використано Vite, який забезпечує швидкий локальний запуск, оновлення сторінок під час розробки та формування статичних файлів для подальшого розгортання [6].

Для маршрутизації між сторінками використовується React Router, який забезпечує організацію переходів між сторінками односторінкового вебзастосунку [7]. Окремо маршрутизація застосовується для підтримки української та англійської мов інтерфейсу через префікси /ua та /en. Захищені маршрути використовуються для сторінок, доступних лише авторизованим користувачам або користувачам із роллю продавця.

Серверну частину реалізовано мовою Python, що забезпечує зручну розробку backend-логіки та інтеграцію з додатковими бібліотеками [8]. Для створення backend API використано Flask, який дає змогу обробляти HTTP-запити та будувати серверну частину вебзастосунку [9]. Backend API приймає HTTP-запити від frontend-частини, виконує автентифікацію користувачів, перевірку ролей, обробку товарів, кошика, замовлень, платежів і відгуків. Для роботи з базою даних використовується SQLAlchemy, що дозволяє описувати сутності системи у вигляді Python-моделей і виконувати операції з даними через ORM-рівень [10].

Для аналізу відгуків використовується AI-модуль класифікації тональності тексту, інтегрований із використанням сервісу Amazon SageMaker [11]. Його завдання полягає у визначенні позитивного або негативного характеру коментаря покупця. Після створення відгуку backend передає текст до модуля аналізу, отримує результат класифікації та зберігає його разом із відгуком. Такий підхід розширює звичайну систему коментарів і демонструє практичне застосування штучного інтелекту в задачах електронної комерції.

Для розгортання застосунку використовується хмарна інфраструктура AWS. Статичні файли frontend-частини можуть розміщуватися в Amazon S3, що дає змогу зберігати зібраний frontend-застосунок як набір HTML, CSS і JavaScript-файлів [12]. Для доставки цих файлів користувачам може використовуватися Amazon CloudFront, який забезпечує швидке поширення контенту та підтримку захищеного доступу через HTTPS [13]. Для backend-частини доцільним є використання EC2 та Nginx як reverse проху, що дозволяє відокремити клієнтську частину від серверної логіки й забезпечити доступ до API у хмарному середовищі.

Отже, обраний набір засобів проєктування забезпечує реалізацію повного циклу роботи інтернет-маркетплейсу: від відображення інтерфейсу користувача до обробки запитів на сервері, збереження даних у SQLite, перевірки ролей користувачів, роботи з відгуками та виконання AI-аналізу їх тональності. Така

технологічна основа є достатньою для навчально-практичного MVP і може бути розширена у разі подальшого розвитку системи.

1.4 Постановка завдання на кваліфікаційну роботу бакалавра

На основі аналізу предметної області, аналогічних програмних рішень і засобів розробки сформульовано завдання на кваліфікаційну роботу бакалавра. Необхідно розробити навчально-практичний інтернет-маркетплейс на основі Python, який забезпечує взаємодію між гостем, покупцем і продавцем, підтримує основні процеси електронної комерції та містить AI-модуль аналізу тональності відгуків.

Розроблювана система повинна підтримувати три основні типи користувачів. Гість має доступ до відкритої частини сайту, може переглядати головну сторінку, каталог і сторінку товару. Покупець після авторизації може додавати товари до кошика, оформлювати замовлення, переглядати власні замовлення та залишати відгуки після підтвердження оплати. Продавець має доступ до seller-панелі, у якій може керувати власними товарами, категоріями та замовленнями, що стосуються його товарних позицій.

Основними вимогами до клієнтської частини є зручний інтерфейс, логічна структура сторінок, адаптивність для мобільних пристроїв, підтримка української та англійської мов, а також коректна взаємодія з серверною частиною через API. Основними вимогами до серверної частини є обробка HTTP-запитів, робота з базою даних, автентифікація користувачів, перевірка ролей, керування товарами, кошиком, замовленнями, оплатою та відгуками.

У межах роботи необхідно реалізувати бізнес-правило, за яким покупець може залишити відгук лише після підтвердження оплати замовлення. Для цього продавець у seller-панелі підтверджує оплату, після чого статус оплати набуває значення paid, а система відкриває покупцю можливість створення відгуку для відповідного товару. Після створення відгуку backend передає текст коментаря

до AI-модуля, отримує результат аналізу тональності та зберігає його разом із відгуком.

Результатом виконання кваліфікаційної роботи має стати функціональний прототип інтернет-маркетплейсу з клієнт-серверною архітектурою, реляційною базою даних, рольовою моделлю доступу, кошиком, системою замовлень, seller-панеллю, механізмом створення відгуків після оплаченої покупки та AI-аналізом їх тональності. Такий прототип повинен демонструвати основні сценарії роботи маркетплейсу й залишати можливість подальшого розширення функціональності.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання

Розроблюваний інтернет-маркетплейс є клієнт-серверною інформаційною системою, у якій клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем, а серверна частина забезпечує обробку запитів, перевірку прав доступу, виконання бізнес-логіки та роботу з базою даних. Такий підхід є доцільним для систем електронної комерції, оскільки дозволяє відокремити інтерфейс користувача від логіки обробки даних і спростити подальшу підтримку програмного продукту.

До основних функціональних вимог системи належать перегляд головної сторінки, каталогу та сторінки окремого товару, реєстрація й авторизація користувача, додавання товарів до кошика, оформлення замовлення, перегляд власних замовлень, робота із seller-панеллю, керування товарами та категоріями, підтвердження оплати замовлення продавцем, створення відгуку після оплаченої покупки та автоматичний аналіз тональності текстового коментаря.

Нефункціональні вимоги до системи охоплюють адаптивність інтерфейсу для desktop- і mobile-пристроїв, підтримку української та англійської мов, розділення frontend- і backend-частин, взаємодію через HTTP API, централізовану перевірку ролей користувачів, збереження даних у реляційній базі даних та можливість подальшого розширення системи. Окремою вимогою є те, що frontend не повинен напряму працювати з базою даних, а має отримувати та передавати дані лише через backend API.

Клієнтська частина системи реалізується як односторінковий вебзастосунок на основі React. Такий підхід дозволяє організувати сторінки у вигляді незалежних компонентів і забезпечити швидкий перехід між ними без повного перезавантаження документа в браузері. Для локальної розробки та збірки frontend-частини використовується Vite, а маршрутизація між головною

сторінкою, каталогом, сторінкою товару, кошиком, checkout, замовленнями та seller-сторінками забезпечується React Router.

Серверна частина реалізується мовою Python із використанням Flask. Backend API приймає HTTP-запити від клієнтської частини, перевіряє стан автентифікації, визначає роль користувача та виконує операції з товарами, кошиком, замовленнями, оплатою, адресами й відгуками. Захищені дії виконуються тільки після перевірки прав доступу: покупець працює з власним кошиком і замовленнями, а продавець має доступ лише до seller-функцій і замовлень, що стосуються його товарів.

Для зберігання даних використовується SQLite у поєднанні з SQLAlchemy. SQLite є зручним рішенням для навчально-практичного прототипу та MVP-рівня розробки, оскільки не потребує окремого серверного процесу й зберігає дані у локальному файлі. SQLAlchemy виконує роль ORM-рівня, який дає змогу описувати таблиці у вигляді моделей Python і виконувати операції з даними через серверну логіку, не надаючи клієнтській частині прямого доступу до бази даних.

Окремим компонентом системи є AI-модуль аналізу відгуків. Після створення відгуку backend передає текст коментаря до модуля аналізу, отримує результат класифікації як positive або negative та зберігає його разом із відгуком. Таким чином, обраний підхід поєднує frontend-застосунок, backend API, реляційну базу даних і AI-модуль у єдину структуру, яка забезпечує основні сценарії роботи інтернет-маркетплейсу та створює основу для подальшого масштабування.

2.2 Функціонально-структурна схема роботи об'єкта проєктування

Функціонально-структурна схема роботи об'єкта проєктування відображає основні компоненти інтернет-маркетплейсу та зв'язки між ними. На етапі проєктування було визначено, що система будується за клієнт-серверним принципом: frontend-застосунок забезпечує взаємодію з користувачем, backend API приймає та обробляє запити, база даних SQLite зберігає інформацію про

користувачів, товари й замовлення, а AI-модуль виконує аналіз тональності відгуків. Така структура дозволяє розділити інтерфейс, бізнес-логіку, збереження даних і інтелектуальну обробку тексту.

На рисунку 2.1 зображено проєктну функціональну схему роботи інтернет-маркетплейсу.

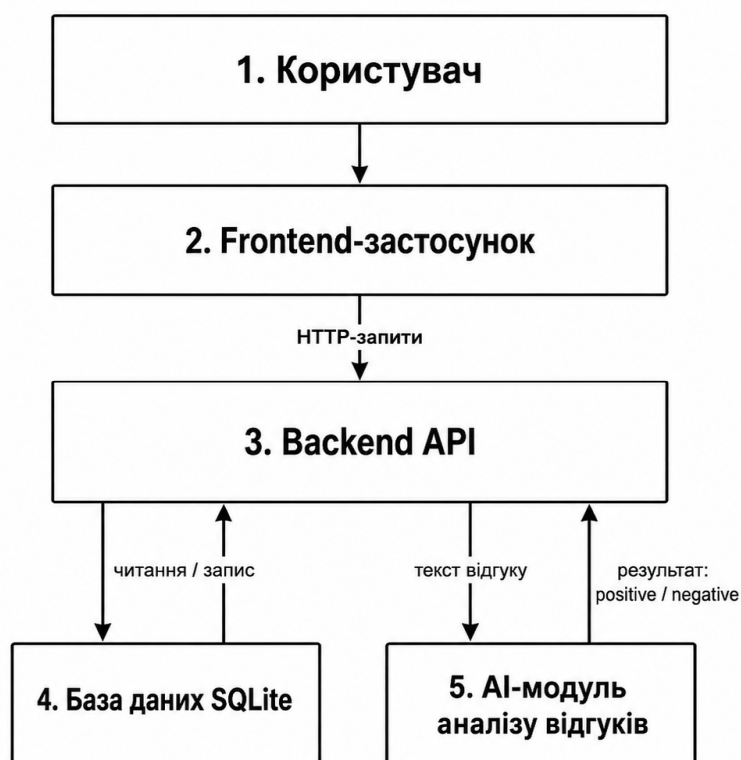


Рисунок 2.1 – Функціональна схема роботи інтернет-маркетплейсу

Джерело: розроблено автором

Користувач взаємодіє із системою через браузер, у якому відкривається frontend-застосунок. Клієнтська частина формує HTTP-запити до backend API, а серверна частина виконує перевірку авторизації, визначає роль користувача та обробляє операції з товарами, кошиком, замовленнями й відгуками. Для збереження даних backend API звертається до SQLite через SQLAlchemy. Під час створення відгуку серверна частина додатково передає текст коментаря до AI-модуля та отримує результат класифікації тональності.

У системі передбачено три основні ролі користувачів: гість, покупець і продавець. Кожна з цих ролей має власний набір доступних дій, тому розмежування прав доступу є одним із ключових елементів проектування інтернет-маркетплейсу. Такий підхід дозволяє відокремити відкриту частину системи від операцій, які потребують авторизації користувача або спеціальних прав продавця. Крім того, рольова модель забезпечує більш контрольовану взаємодію з даними, оскільки користувач отримує доступ лише до тих функцій, які відповідають його ролі в системі.

Гість є неавторизованим користувачем системи. Він має доступ до відкритої частини маркетплейсу та може виконувати базові дії, пов'язані з ознайомленням із товарами. Зокрема, гість може переглядати головну сторінку, переходити до каталогу товарів, відкривати сторінки окремих товарів, переглядати їх описи, ціни, залишки на складі та відгуки інших користувачів. Такий рівень доступу є достатнім для початкового ознайомлення з можливостями маркетплейсу та вибору товару, який потенційно може зацікавити користувача.

Водночас дії, пов'язані з оформленням замовлення або створенням відгуку, для гостя недоступні. Аналогічно створення відгуку має бути доступним лише тим користувачам, які фактично здійснили покупку. У разі спроби виконати таку дію система спрямовує гостя до сторінки авторизації або реєстрації.

На рисунку 2.2 подано сценарій взаємодії гостя з інтернет-маркетплейсом. Діаграма відображає послідовність дій неавторизованого користувача: відкриття сайту, перегляд головної сторінки, перехід до каталогу, вибір товару та ознайомлення з інформацією про нього. Цей сценарій визначає межі доступу гостя і показує, що відкриті функції системи призначені насамперед для перегляду інформації, а купівля товарів та створення відгуків потребують входу в систему.

Покупець є авторизованим користувачем, який має доступ до основного процесу придбання товару. Після входу в систему він може переглядати каталог, відкривати сторінки товарів, додавати товари до кошика, перевіряти вибрані

позиції, вказувати або вибирати адресу доставки, оформлювати замовлення та переглядати його поточний стан. На відміну від гостя, покупець взаємодіє не лише з відкритою частиною маркетплейсу, а й із персоналізованими даними, зокрема власним кошиком, замовленнями та історією покупок.

Додатковою можливістю покупця є створення відгуку про товар. Покупець може залишити відгук тільки тоді, коли для відповідного замовлення було підтверджено оплату і статус оплати набув значення paid.

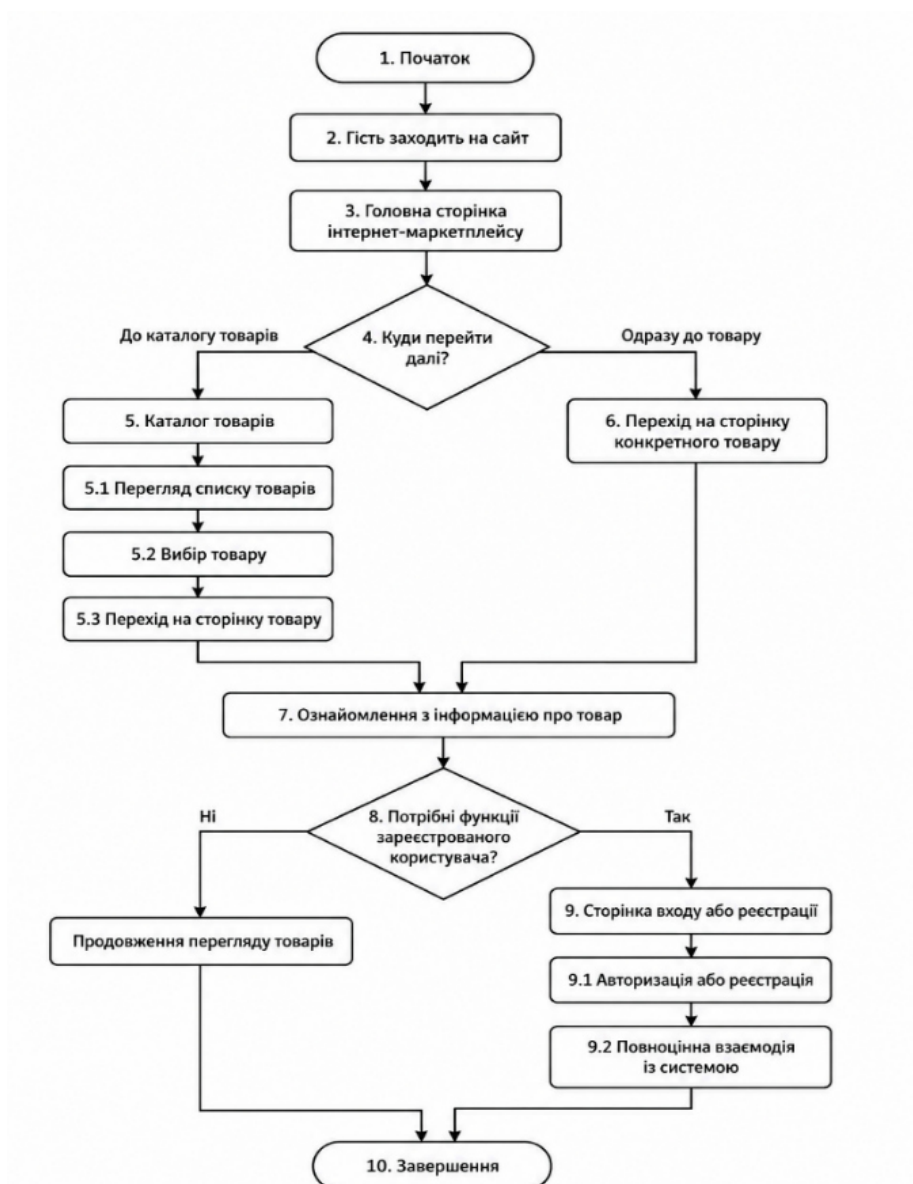


Рисунок 2.2 – Діаграма сценарію взаємодії гостя з маркетплейсом

Джерело: розроблено автором

На рисунку 2.3 зображено сценарій взаємодії покупця з маркетплейсом. Діаграма демонструє послідовність дій від перегляду товарів до оформлення замовлення.

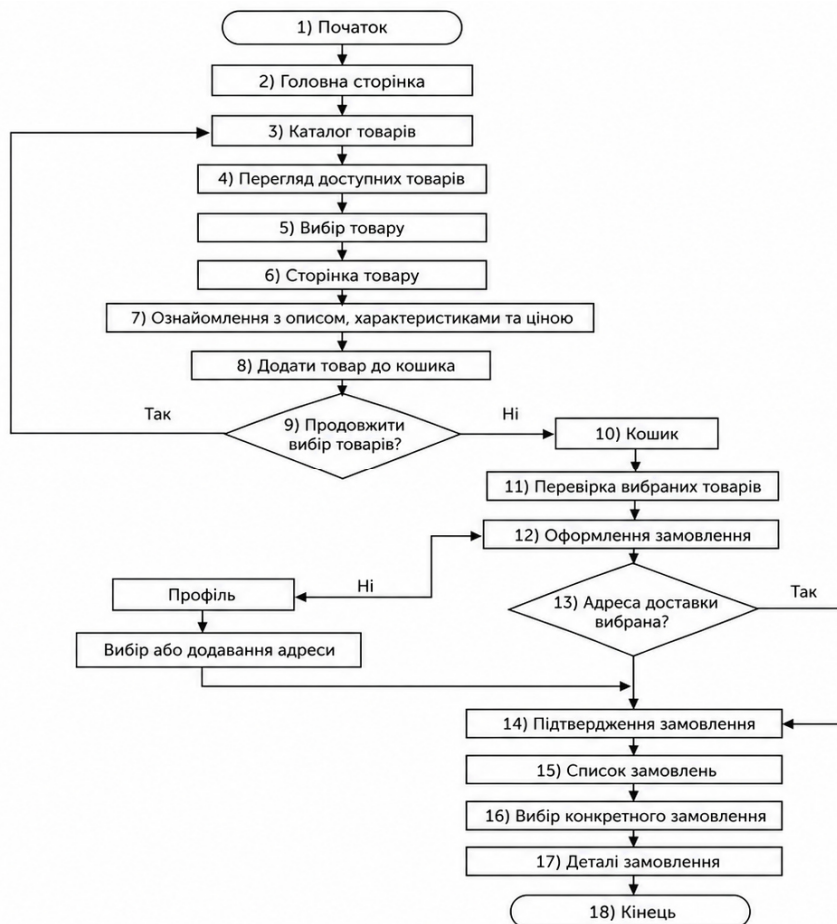


Рисунок 2.3 – Діаграма сценарію взаємодії покупця з маркетплейсом

Джерело: розроблено автором

Покупець обирає товар, додає його до кошика, переходить до checkout, перевіряє дані замовлення та підтверджує покупку. Після створення замовлення система зберігає його у базі даних із початковим статусом pending. Надалі покупець може переглядати деталі замовлення та, після підтвердження оплати продавцем, залишити відгук про придбаний товар.

Продавець є авторизованим користувачем, який має доступ до окремої seller-частини системи. Через панель продавця він може переглядати власні товари, створювати нові товарні позиції, редагувати інформацію про товари,

працювати з категоріями, переглядати замовлення, що стосуються його товарів, і підтверджувати оплату таких замовлень. У межах прототипу підтвердження оплати означає, що статус запису Payment набуває значення paid, після чого покупець отримує право залишити відгук.

На рисунку 2.4 наведено сценарій взаємодії продавця з маркетплейсом.

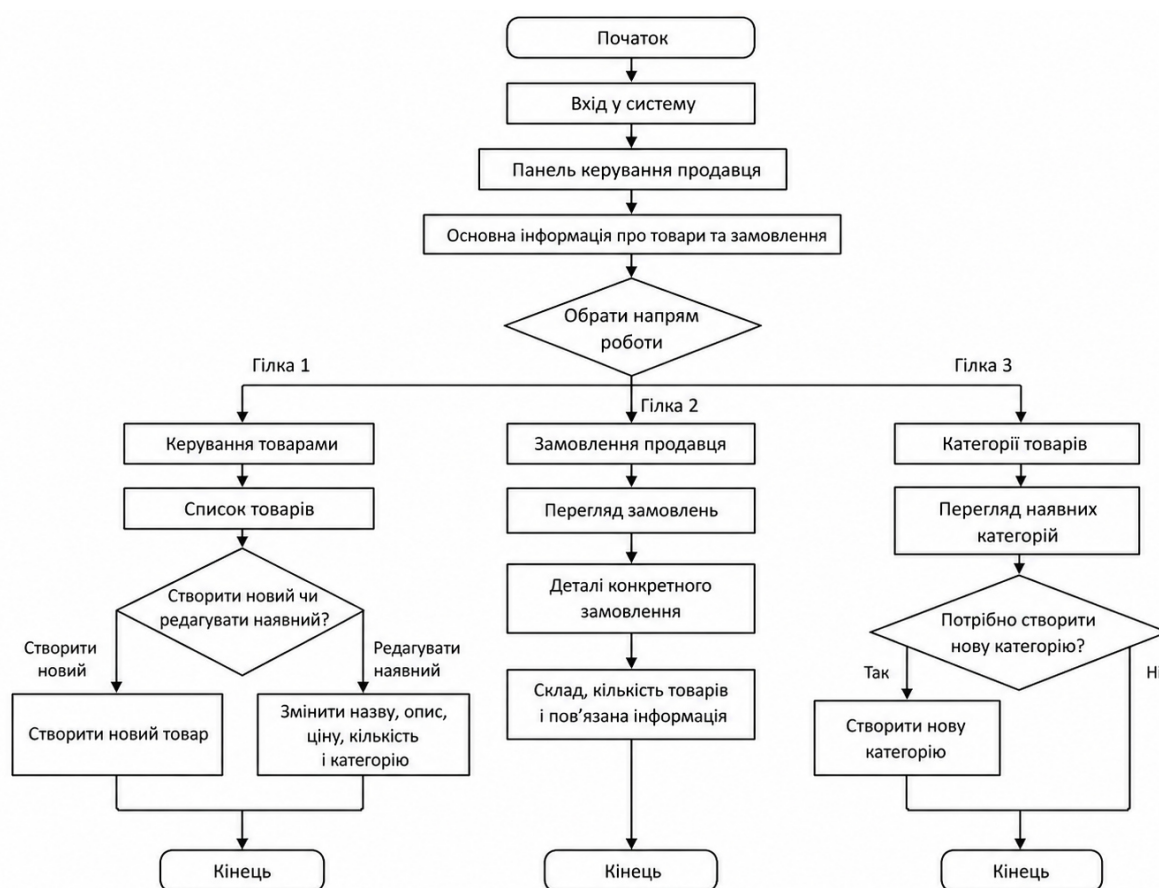


Рисунок 2.4 – Діаграма сценарію взаємодії продавця з маркетплейсом

Джерело: розроблено автором

Діаграма відображає основні напрями роботи seller-панелі: керування товарами, перегляд замовлень продавця, редагування категорій і підтвердження оплати замовлення. Наявність окремого сценарію для продавця підтверджує, що система реалізує не лише функції звичайного інтернет-магазину, а й базову логіку маркетплейсу з розподілом ролей між покупцем і продавцем.

Для уточнення логіки виконання основних процесів системи було розроблено блок-схеми алгоритмів. Вони відображають послідовність дій під час

оформлення замовлення, авторизації користувача та створення відгуку з подальшим AI-аналізом. Такі блок-схеми потрібні для формалізації основних сценаріїв до етапу програмної реалізації.

На рисунку 2.5 подано алгоритм оформлення замовлення. Процес починається з відкриття кошика користувачем. Якщо кошик порожній, користувач повертається до каталогу товарів. Якщо в кошику є товари, система переводить користувача до сторінки оформлення замовлення, де перевіряється адреса доставки та підтверджуються дані покупки. Після підтвердження backend створює запис замовлення у базі даних, формує пов'язані позиції замовлення та встановлює початковий статус pending

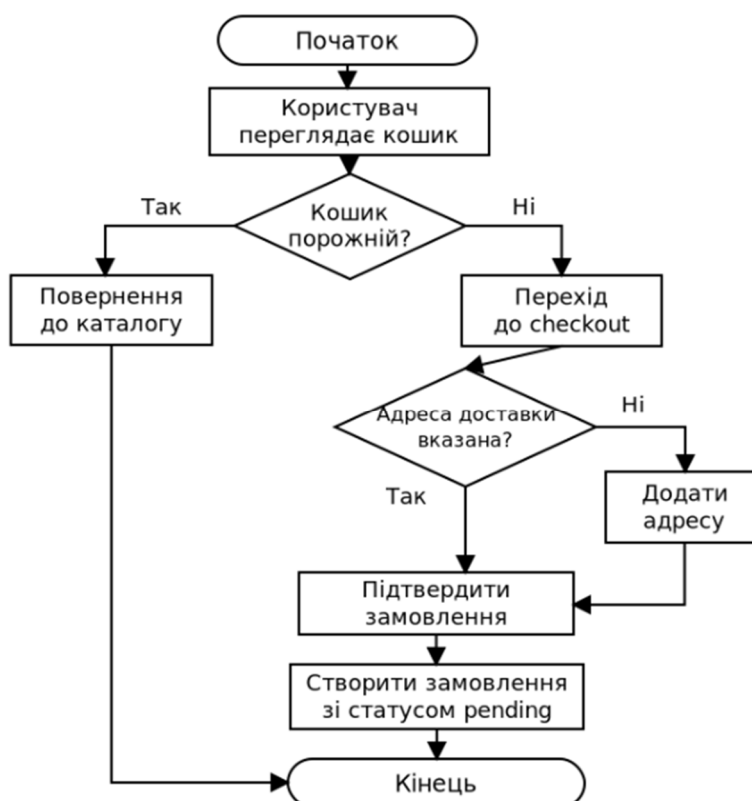


Рисунок 2.5 – Блок-схема алгоритму оформлення замовлення

Джерело: розроблено автором

На рисунку 2.6 зображено алгоритм авторизації користувача. Користувач вводить email і пароль, після чого дані передаються до backend API. Серверна частина перевіряє наявність облікового запису та правильність введеного пароля.

У разі помилки система повертає повідомлення про невдалу авторизацію. Якщо перевірка успішна, створюється користувацька сесія, визначається роль користувача, а подальший доступ до сторінок і API-маршрутів надається відповідно до цієї ролі.

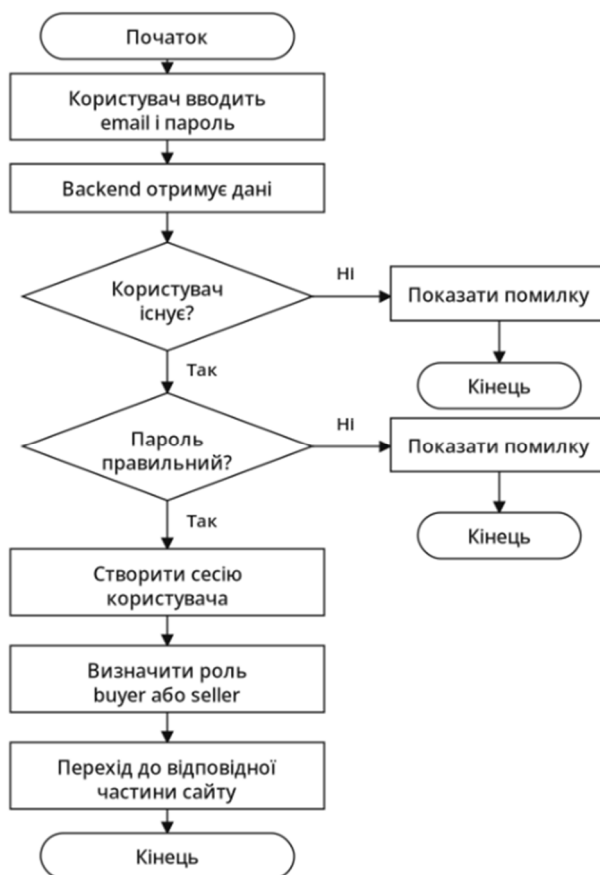


Рисунок 2.6 – Блок-схема алгоритму авторизації користувача

Джерело: розроблено автором

На рисунку 2.7 наведено алгоритм створення відгуку та його AI-аналізу. Спочатку покупець відкриває деталі замовлення або сторінку товару. Система перевіряє, чи є користувач авторизованим покупцем, чи не залишав він відгук раніше та чи має він оплачений запис замовлення з відповідним товаром. Якщо статус оплати не дорівнює paid, форма відгуку залишається недоступною. Якщо умова виконана, покупець вводить текст коментаря та надсилає його до системи. Backend зберігає відгук, передає текст до AI-модуля аналізу тональності, отримує результат класифікації та зберігає його разом із відгуком.



Рисунок 2.7 – Блок-схема алгоритму створення та AI-аналізу відгуку

Джерело: розроблено автором

Для узагальнення структури програмного забезпечення було побудовано UML-діаграму класів. Вона відображає основні сутності предметної області, їх призначення та зв'язки, які надалі використовуються під час створення моделей бази даних.

На рисунку 2.8 подано UML-діаграму класів інтернет-маркетплейсу. Центальною сутністю є User, яка описує користувача системи та пов'язана з роллю, кошиком, замовленнями та відгуками. SellerProfile використовується для зберігання додаткової інформації про продавця. Product описує товар і пов'язаний із категорією, продавцем, зображеннями, елементами кошика, елементами замовлення та відгуками. CartItem і OrderItem зберігають окремі позиції у кошику та замовленні. Така модель показує, як користувачі, товари, замовлення, платежі й відгуки взаємодіють у межах однієї системи.

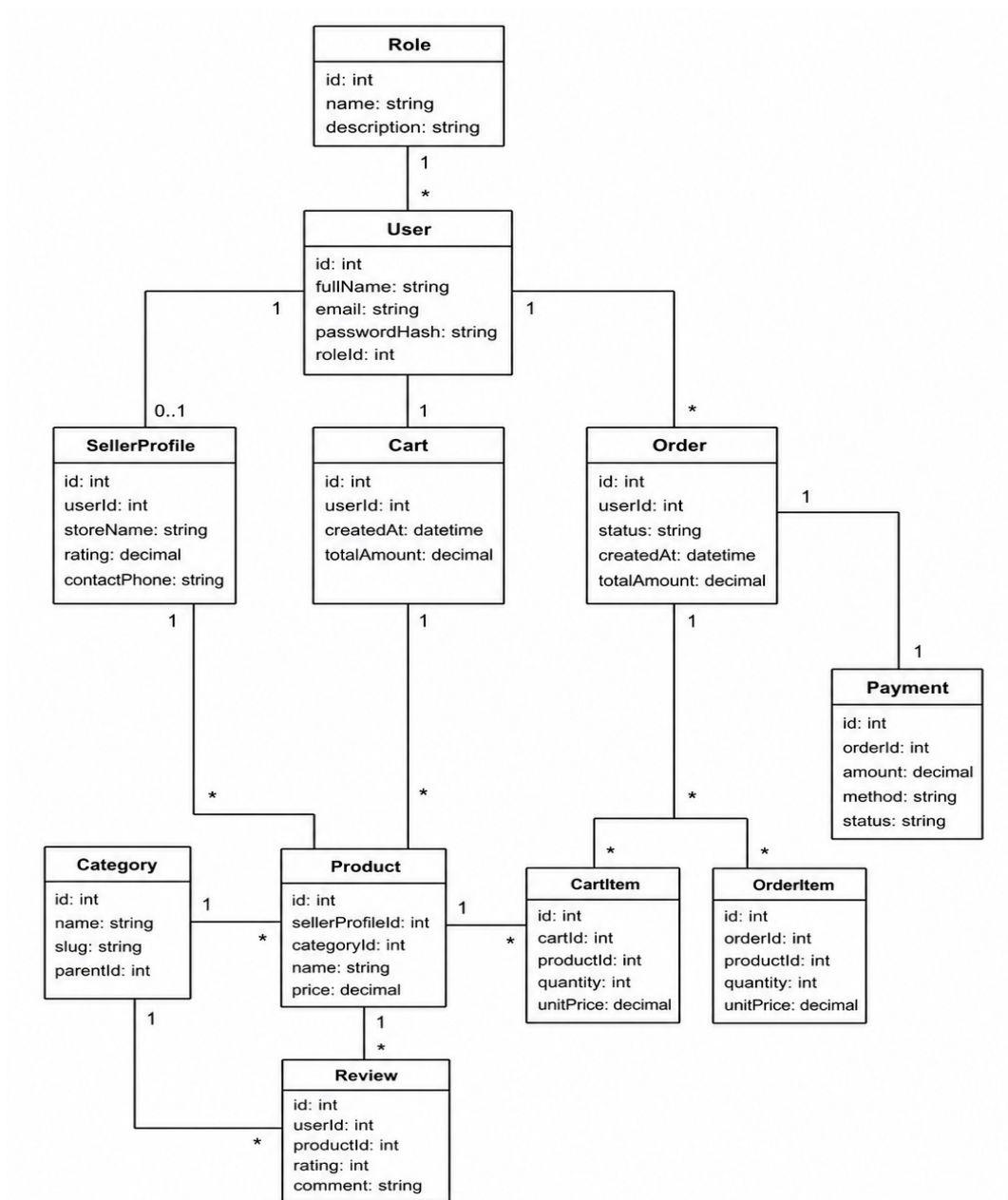


Рисунок 2.8 – UML-діаграма класів інтернет-маркетплейсу

Джерело: розроблено автором

Окрім діаграми класів, було побудовано UML-діаграму компонентів, яка показує логічну структуру системи на рівні основних програмних модулів і каналів взаємодії між ними.

На рисунку 2.9 зображено компонентну структуру інтернет-маркетплейсу. Frontend React App забезпечує користувацький інтерфейс і взаємодіє з API Client, який формує запити до Flask Backend API. Серверна частина містить логічні модулі Auth Module, Catalog Module, Cart and Orders Module, Seller Module та

Reviews Module. Для зберігання даних використовується SQLite Database. Окремо виділено AI Review Analysis Module, який отримує текст відгуку з Reviews Module і повертає результат аналізу тональності. Така структура підкреслює, що frontend не звертається до бази даних або AI-модуля напряму, а працює з ними через backend API.

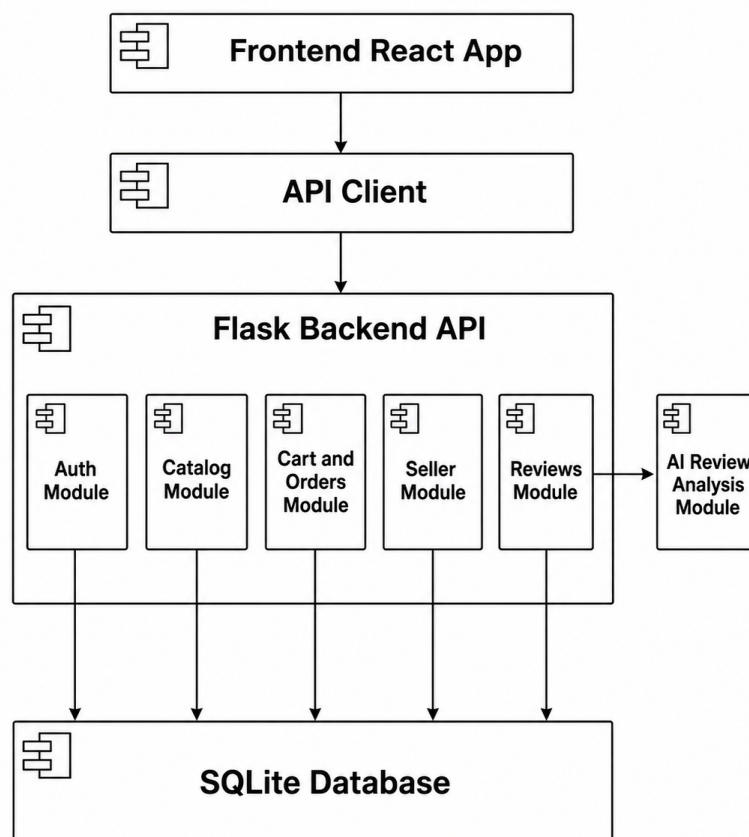


Рисунок 2.9 – UML-діаграма компонентів інтернет-маркетплейсу

Джерело: розроблено автором

Таким чином, у межах функціонально-структурного проектування було визначено основні компоненти інтернет-маркетплейсу, сценарії взаємодії користувачів із системою, алгоритми виконання ключових операцій та логічну структуру програмного забезпечення. Побудовані схеми підтверджують, що система має клієнт-серверну архітектуру, підтримує різні ролі користувачів, забезпечує роботу з товарами, кошиком, замовленнями, оплатою, відгуками та містить окремий AI-модуль для аналізу текстових коментарів.

2.3 Створення моделі бази даних

Проектування бази даних є важливим етапом розробки інтернет-маркетплейсу, оскільки саме база даних забезпечує зберігання основної інформації про користувачів, товари, замовлення, кошик, категорії, оплату та відгуки. Сучасні системи керування базами даних можна умовно поділити на реляційні, документоорієнтовані, графові та key-value системи. Для розроблюваного застосунку було обрано реляційну модель даних, оскільки предметна область маркетплейсу має чітко визначені сутності та зв'язки між ними.

У системі використовується база даних SQLite, яка не потребує окремого серверного процесу та зберігає дані у локальному файлі [14]. Такий вибір є доцільним для навчально-практичного проєкту та MVP-рівня розробки, оскільки SQLite не потребує окремого серверного процесу, зберігає дані у локальному файлі та забезпечує достатню функціональність для реалізації основних сценаріїв роботи маркетплейсу. Для взаємодії з базою даних використовується SQLAlchemy, який виконує роль ORM-рівня між Python-кодом і таблицями бази даних. Це дозволяє описувати сутності у вигляді моделей і виконувати операції створення, читання, оновлення та видалення даних без написання сирих SQL-запитів у кожному фрагменті бізнес-логіки.

Основними сутностями бази даних є User, Role, SellerProfile, Address, Category, Product, ProductImage, Cart, CartItem, Order, OrderItem, Payment та Review. Така структура дозволяє відобразити всі ключові процеси системи: реєстрацію користувачів, розподіл ролей, створення товарів, роботу з кошиком, оформлення замовлень, зберігання адрес доставки, облік позицій замовлення, моделювання оплати та створення відгуків з AI-аналізом тональності.

Сутність User призначена для зберігання інформації про користувачів системи. Вона містить дані, необхідні для автентифікації та ідентифікації користувача, зокрема email, хешований пароль, стан активності та зв'язок із роллю. Через роль визначається, які функції доступні користувачу: перегляд

відкритих сторінок, оформлення замовлень, залишення відгуків або керування товарами в seller-панелі.

Сутність `Role` використовується для розмежування прав доступу. У системі передбачено ролі, які визначають поведінку користувача в застосунку: покупець може працювати з кошиком, замовленнями та відгуками, а продавець має доступ до функцій керування товарами й обробки замовлень, пов'язаних із його товарними позиціями. Гість не потребує окремого запису в таблиці ролей, оскільки є неавторизованим користувачем.

Для продавця використовується додаткова сутність `SellerProfile`. Вона містить інформацію про профіль продавця, зокрема назву магазину або інші дані, необхідні для представлення продавця в системі. Такий підхід дозволяє відокремити загальні дані користувача від інформації, яка стосується саме продавця та його storefront.

Сутність `Address` використовується для зберігання адрес доставки покупця. Один користувач може мати декілька адрес, що дозволяє вибирати потрібну адресу під час оформлення замовлення. Наявність окремої таблиці адрес спрощує повторне використання даних і робить процес checkout зручнішим для користувача.

Сутність `Category` призначена для групування товарів у каталозі. Категорії дозволяють структурувати товари за типами або призначенням. У разі потреби структура категорій може бути розширена за рахунок використання батьківських категорій, що дозволяє створювати ієрархію розділів каталогу.

Сутність `Product` описує товар, який додається продавцем до маркетплейсу. Вона містить назву, опис, ціну, залишок на складі, зв'язок із категорією та зв'язок із продавцем. Така структура дозволяє відображати товари в каталозі, відкривати сторінку окремого товару, додавати товар до кошика та використовувати його у складі замовлень.

Для зберігання зображень товарів використовується сутність `ProductImage`. Один товар може мати декілька зображень, тому між `Product` і `ProductImage`

встановлюється зв'язок один-до-багатьох. Це дозволяє відображати основне та додаткові зображення товару на сторінці його детального перегляду.

Кошик покупця реалізовано за допомогою сутностей Cart і CartItem. Сутність Cart пов'язана з конкретним користувачем і зберігає загальну інформацію про кошик. Сутність CartItem описує окрему позицію в кошику, тобто зв'язує товар із кошиком і зберігає кількість вибраного товару. Завдяки цьому користувач може додавати товари до кошика, змінювати їх кількість або видаляти окремі позиції.

Після підтвердження покупки на основі вмісту кошика створюється замовлення. Сутність Order містить інформацію про покупця, дату створення, статус замовлення, загальну суму та адресу доставки. Початковий статус замовлення встановлюється як pending. Надалі продавець не змінює саме замовлення на paid, а підтверджує оплату через пов'язаний запис Payment; у межах прототипу після цього замовлення може вважатися виконаним або доставленим.

Сутність OrderItem використовується для зберігання окремих позицій у замовленні. Вона пов'язує замовлення з конкретними товарами, фіксує їх кількість і ціну на момент оформлення.

Сутність Payment використовується для фіксації інформації про оплату замовлення. У межах розроблюваного прототипу онлайн-оплата не реалізується як окремий платіжний сервіс, тому оплата моделюється через зміну статусу запису Payment, пов'язаного із замовленням.

Сутність Review призначена для зберігання відгуків покупців про товари. Відгук пов'язаний із користувачем і товаром, містить текст коментаря, оцінку та результат AI-аналізу тональності.

На рисунку 2.10 наведено фізичну структуру бази даних інтернет-маркетплейсу. Схема демонструє основні таблиці системи та зв'язки між ними. Центральне місце займає таблиця users, яка пов'язана з ролями, адресами, кошиком, замовленнями та відгуками. Таблиця products пов'язана з категоріями, продавцями, зображеннями, позиціями кошика, позиціями замовлень і

відгуками. Окремо представлено таблицю payments, яка зберігає стан оплати замовлення. Така структура забезпечує цілісне представлення основних бізнес-процесів маркетплейсу на рівні бази даних.

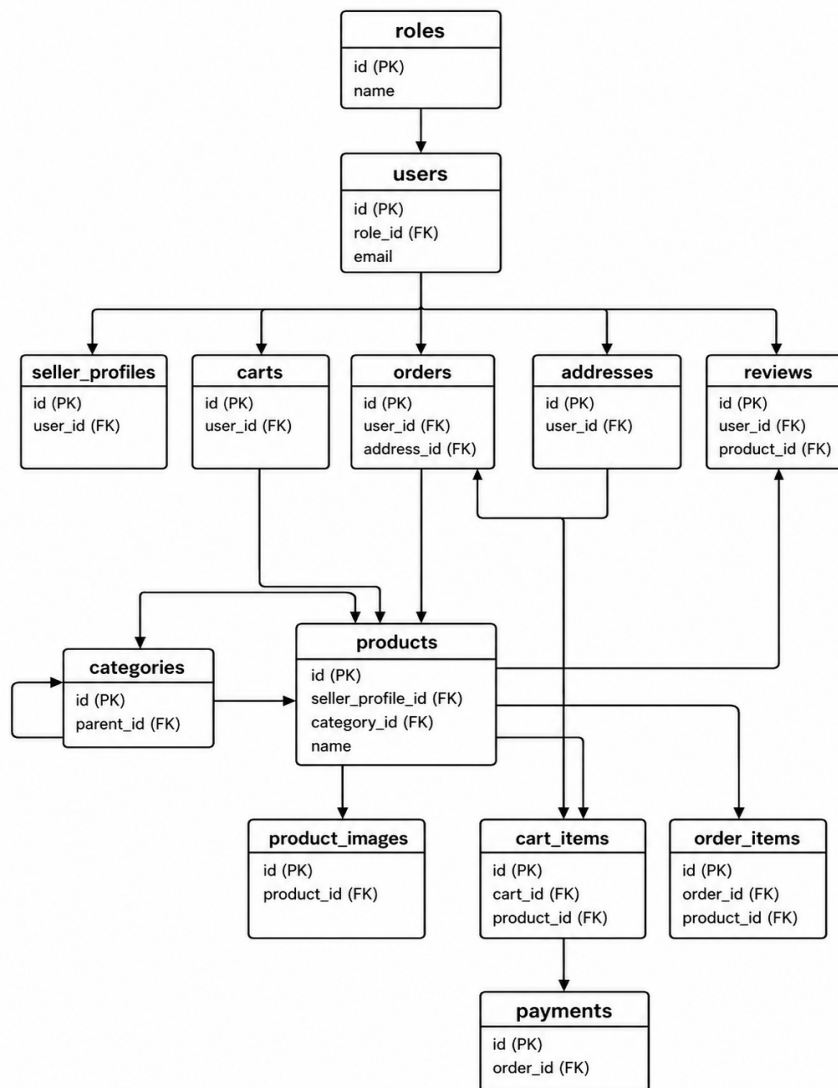


Рисунок 2.10 – Фізична структура бази даних

Джерело: розроблено автором

Окрему роль у реалізації системи відіграє SQLite. Ця база даних є файловою, тому всі дані зберігаються в одному локальному файлі. Вона не потребує встановлення окремого сервера баз даних, що спрощує розгортання та тестування застосунку. SQLite добре підходить для прототипів, навчальних проєктів і систем з помірним обсягом даних.

Взаємодія застосунку з базою даних організована через backend-частину. Frontend не має прямого доступу до SQLite і не виконує запити до бази даних самостійно. Клієнтська частина лише формує HTTP-запити до backend API. Серверна частина приймає ці запити, виконує перевірку доступу, обробляє бізнес-логіку та через SQLAlchemy звертається до відповідних таблиць бази даних. Такий підхід зменшує ризик неконтрольованої зміни даних

На рисунку 2.11 показано схему взаємодії застосунку з базою даних. Запит користувача спочатку формується у frontend-частині, після чого передається як HTTP API request до Flask route. Далі серверна частина використовує SQLAlchemy для виконання операцій читання або запису в SQLite database. Після обробки запиту результат повертається у зворотному напрямку: від бази даних до SQLAlchemy, потім до Flask route, backend API і frontend-застосунку.

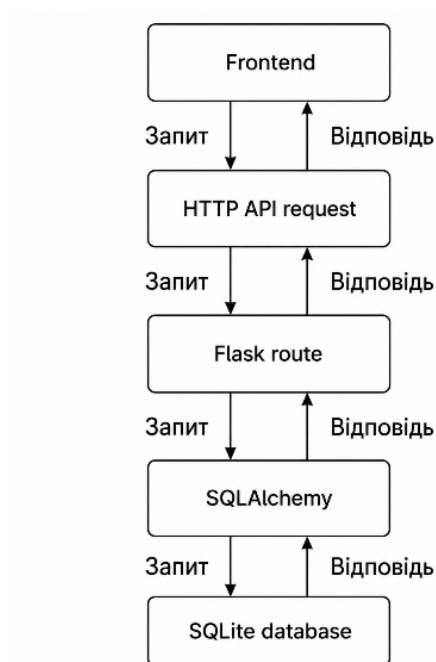


Рисунок 2.11 – Схема взаємодії застосунку з базою даних

Джерело: розроблено автором

Таким чином, розроблена модель бази даних забезпечує зберігання всіх основних даних, необхідних для роботи інтернет-маркетплейсу. Вона відображає зв'язки між користувачами, ролями, товарами, категоріями, кошиком, замовленнями, оплатою та відгуками. Використання SQLite у поєднанні з

SQLAlchemy дозволяє реалізувати просту, зрозумілу та достатньо гнучку структуру даних для навчально-практичного вебзастосунку, а також залишає можливість подальшого переходу на потужнішу реляційну СУБД.

РОЗДІЛ 3

РОЗРОБКА ІНТЕРНЕТ-МАРКЕТПЛЕЙСУ

3.1 Практична реалізація об'єкта проєктування. Побудова блок-схем модулів програми

Архітектура реалізованого інтернет-маркетплейсу побудована за клієнт-серверним принципом. Клієнтська частина відповідає за відображення інтерфейсу користувача та взаємодію з основними сторінками застосунку. Серверна частина забезпечує обробку HTTP-запитів, автентифікацію користувачів, перевірку ролей, виконання бізнес-логіки та роботу з базою даних. Для збереження інформації використовується SQLite, а для аналізу текстових відгуків застосовується AI-модуль класифікації тональності.

На рисунку 3.1 зображено загальну архітектуру реалізованого інтернет-маркетплейсу. Користувач взаємодіє із застосунком через браузер, у якому відкривається frontend-частина. Frontend надсилає HTTP-запити до backend API, який обробляє авторизацію, товари, кошик, замовлення, seller-функції та відгуки.

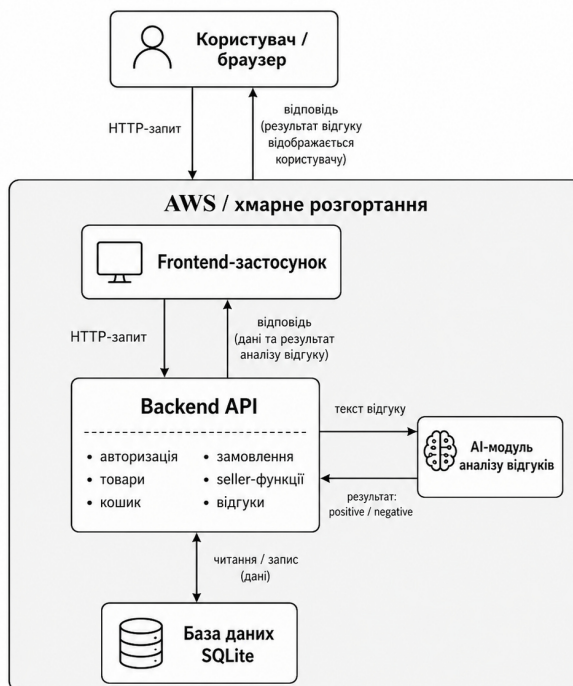


Рисунок 3.1 – Загальна архітектура реалізованого інтернет-маркетплейсу

Джерело: розроблено автором

Для збереження даних backend звертається до SQLite через SQLAlchemy. Під час створення відгуку backend додатково передає текст коментаря до AI-модуля, який повертає результат аналізу тональності.

Одним із базових елементів frontend-частини є маршрутизація. Вона визначає, які сторінки доступні користувачу, які маршрути є відкритими, а які потребують авторизації або наявності ролі продавця. Відкритими сторінками є головна сторінка, каталог і сторінка товару. Кошик, checkout і замовлення доступні лише авторизованому користувачу, а seller-панель доступна тільки користувачу з роллю продавця.

У лістингу 3.1 наведено фрагмент конфігурації маршрутів клієнтської частини. Він демонструє підтримку мовних шляхів /ua та /en, відкритих сторінок каталогу і товарів, а також захищених маршрутів для кошика, checkout, замовлень і seller-панелі.

Лістинг 3.1 – Конфігурація маршрутів клієнтської частини

```
export default function App() {
  return (
    <Routes>
      <Route path="/" element={<Navigate
to={withLang(DEFAULT_LANG)} replace />} />
      <Route path=":lang" element={<LanguageProvider><Layout
/></LanguageProvider>}>
        <Route index element={<HomePage />} />
        <Route path="catalog" element={<CatalogPage />} />
        <Route path="products/:productId" element={<ProductPage
/>} />
        <Route path="cart" element={<RequireAuth><CartPage
/></RequireAuth>} />
        <Route path="checkout" element={<RequireAuth><CheckoutPage
/></RequireAuth>} />
        <Route path="orders" element={<RequireAuth><OrdersPage
/></RequireAuth>} />
        <Route path="orders/:orderId"
element={<RequireAuth><OrderDetailsPage /></RequireAuth>} />
        <Route path="seller" element={<RequireAuth
role="seller"><SellerDashboardPage /></RequireAuth>} />
        <Route path="seller/products" element={<RequireAuth
role="seller"><SellerProductsPage /></RequireAuth>} />
        <Route path="seller/orders/:orderId" element={<RequireAuth
role="seller"><SellerOrderDetailsPage /></RequireAuth>} />
      </Route>
    </Routes>
  )
}
```

```

    <Route path="*" element={<Navigate
to={withLang(DEFAULT_LANG)} replace />} />
  </Routes>
);
}

```

кінець лістингу 3.1

Компонент `RequireAuth` використовується для обмеження доступу до сторінок, які потребують авторизації, а параметр `role="seller"` додатково перевіряє право користувача на доступ до seller-функцій.

На рисунку 3.2 показано мобільну навігацію інтернет-маркетплейсу. Для малих екранів реалізовано кнопку з трьома смужками, яка відкриває бічну панель меню.

Каталог товарів є однією з основних сторінок маркетплейсу, оскільки саме з нього користувач починає пошук потрібної товарної позиції. На цій сторінці відображаються доступні товари, їх ціни, короткі характеристики та можливість переходу до детального перегляду.

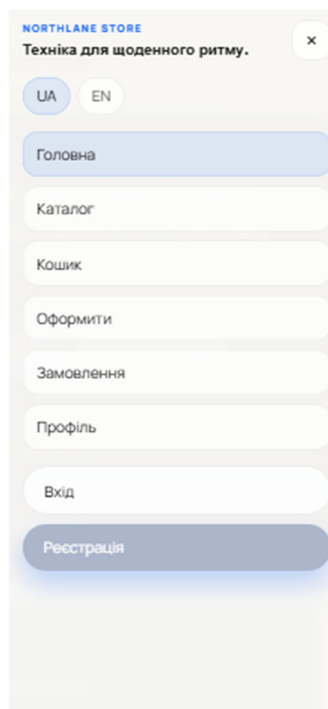


Рисунок 3.2 – Мобільна навігація інтернет-маркетплейсу

Джерело: розроблено автором

У лістингу 3.2 показано реалізацію адаптивної навігації. Стан `isMobileMenuOpen` відповідає за відкриття або закриття мобільної панелі. Обробник натискання змінює стан меню, а `useEffect` автоматично закриває його після переходу на іншу сторінку. Це робить навігацію зручною для мобільних пристроїв і забезпечує коректну поведінку інтерфейсу під час переходів між сторінками.

Лістинг 3.2 – Реалізація адаптивної навігації

```
const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

useEffect(() => {
  setIsMobileMenuOpen(false);
}, [location.pathname]);

<button
  type="button"
  className={`mobile-menu-button${isMobileMenuOpen ? " active" :
""}`}
  aria-expanded={isMobileMenuOpen}
  onClick={() => setIsMobileMenuOpen((current) => !current)}
>
  <span />
  <span />
  <span />
</button>

<aside className={`mobile-drawer${isMobileMenuOpen ? " open" :
""}`}>
  <nav className="mobile-drawer-nav">
    {visibleNavItems.map((item) => (
      <NavLink key={item.to} to={item.to} className="mobile-
drawer-link">
        {item.label}
      </NavLink>
    ))}
  </nav>
</aside>
```

кінець лістингу 3.2

На рисунку 3.3 наведено сторінку каталогу товарів. Вона містить список доступних товарів, що завантажуються з backend API. Користувач може переглядати основну інформацію про товари, фільтрувати їх за категоріями та переходити до сторінки детального перегляду окремої товарної позиції.

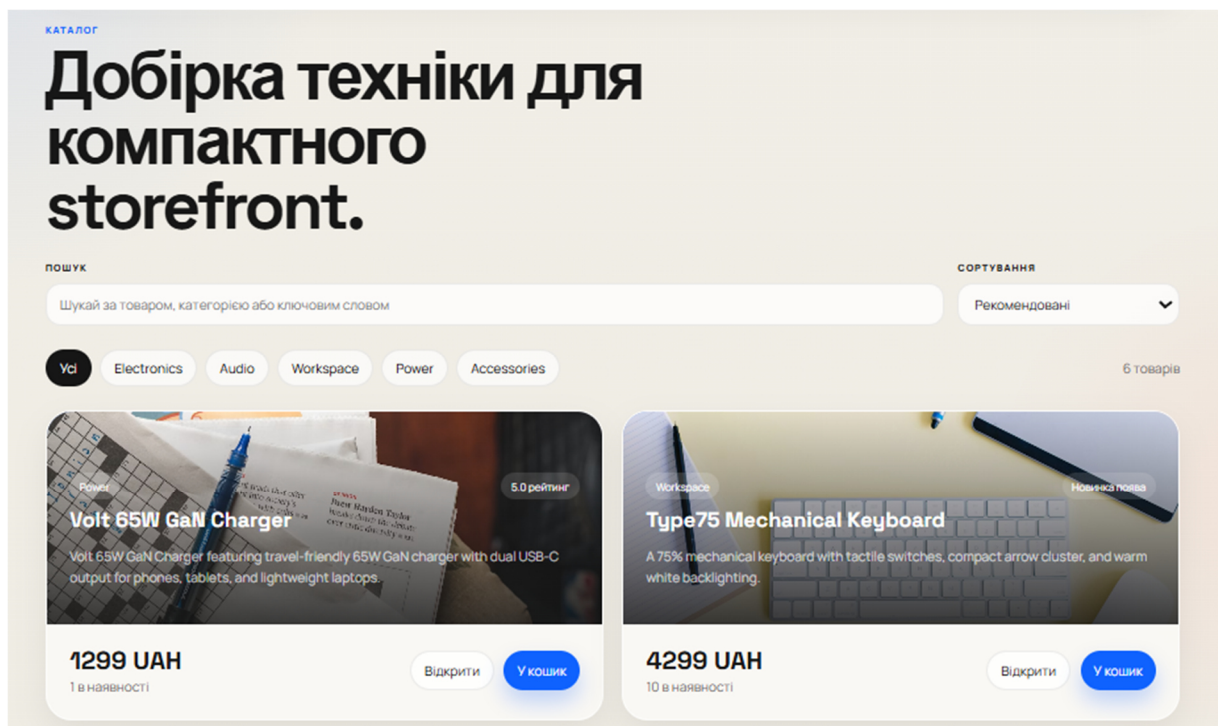


Рисунок 3.3 – Сторінка каталогу товарів

Джерело: розроблено автором

Сторінка окремого товару використовується для відображення повної інформації про вибраний товар. Вона містить назву, опис, ціну, залишок на складі, зображення, кнопку додавання до кошика та блок відгуків.

У лістингу 3.3 наведено фрагмент отримання та підготовки товарів для відображення в каталозі. Дані про товари та категорії завантажуються асинхронно з backend API. Використання Promise.all дозволяє одночасно отримати обидва набори даних, а useMemo застосовується для фільтрації та сортування товарів перед відображенням у компоненті.

Лістинг 3.3 – Отримання та відображення товарів у каталозі

```
useEffect(() => {
  async function loadData() {
    try {
      setLoading(true);
      const [productsData, categoriesData] = await Promise.all([
        fetchProducts(),
        fetchCategories(),
      ]);
      setProducts(productsData);
      setCategories(categoriesData);
    }
  }
  loadData();
});
```

```

    } catch (loadError) {
      setError(loadError.message);
    } finally {
      setLoading(false);
    }
  }
  loadData();
}, []);
const filteredProducts = useMemo(() => {
  let nextProducts = products;
  if (selectedCategory !== "all") {
    nextProducts = nextProducts.filter(
      (product) => String(product.category?.id) ===
selectedCategory,
    );
  }
  return [...nextProducts].sort((a, b) => b.average_rating -
a.average_rating);
}, [products, selectedCategory]);

```

кінець лістингу 3.3

На рисунку 3.4 показано сторінку окремого товару. Вона дозволяє користувачу ознайомитися з детальною інформацією про товар і додати його до кошика. Також на сторінці відображаються відгуки покупців, що дає змогу оцінити досвід інших користувачів перед оформленням замовлення.

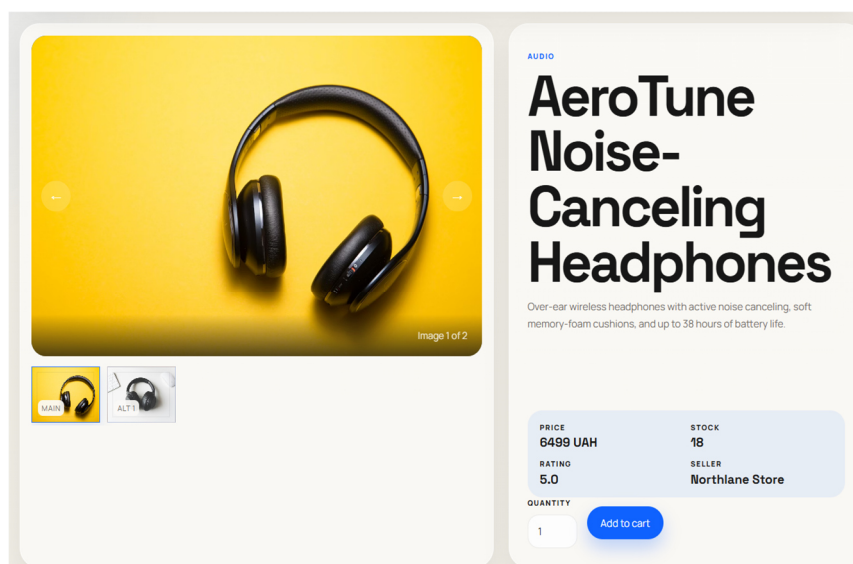


Рисунок 3.4 – Сторінка окремого товару з відгуками

Джерело: розроблено автором

Оформлення замовлення є одним із ключових сценаріїв роботи покупця. На цьому етапі користувач перевіряє вибрані товари, загальну суму, адресу доставки та підтверджує створення замовлення.

У лістингу 3.4 показано механізм додавання товару до кошика. Якщо користувач не авторизований, система спрямовує його на сторінку входу. Якщо користувач увійшов у систему, frontend надсилає POST-запит до backend API з ідентифікатором товару та кількістю. Такий підхід забезпечує обмеження доступу до функції купівлі для неавторизованих користувачів.

Лістинг 3.4 – Додавання товару до кошика

```

async function handleAddToCart() {
  if (!product) {
    return;
  }
  if (!isAuthenticated) {
    navigate(path("/login"), { state: { from: location } });
    return;
  }

  try {
    await addCartItem(product.id, quantity);
    setMessage(`${quantity} item${quantity === 1 ? "" : "s"} added
to cart.`);
  } catch (cartError) {
    setMessage(cartError.message);
  }
}

export async function addCartItem(productId, quantity = 1) {
  const response = await jsonRequest("/api/me/cart/items", {
    method: "POST",
    body: JSON.stringify({ product_id: productId, quantity }),
  });
  return readJson(response);
}

```

кінець лістингу 3.4

На рисунку 3.5 зображено сторінку оформлення замовлення. На ній покупець перевіряє вміст кошика, загальну суму та дані доставки. Після підтвердження backend створює замовлення у базі даних, формує позиції замовлення та пов'язаний запис оплати.

Окрема seller-панель реалізована для користувачів із роллю продавця. Вона дозволяє відокремити функції керування товарами та замовленнями від звичайних дій покупця.

Перевір адресу, підтвердь оплату, створи замовлення

КОНТАКТ
oleg12267@gmail.com
 Цей акаунт буде використано для оновлень замовлення, зв'язку по доставці та історії покупок.

АДРЕСА ДОСТАВКИ
Обери, куди має поїхати замовлення
☒ Kamin-kashyrsky, Україна
 volia 21, 44500

ОПЛАТА
Обери спосіб оплати
☒ Карткою онлайн
☐ Готівкою при отриманні
☐ Банківський переказ

КОМЕНТАР ДО ЗАМОВЛЕННЯ
 Необов'язковий коментар до доставки

ПІДСУМОК ЗАМОВЛЕННЯ
6499 UAH
 Кількість 1
 Товари 1
 Оплата Карткою онлайн
 Коментар Немає
 AeroTune Noise-Cancelling Headphones 1 x 6499 UAH
 доставити сюди
 Kamin-kashyrsky, Україна
 volia 21, 44500
 Створити замовлення Назад до кошика

Рисунок 3.5 – Сторінка оформлення замовлення

Джерело: розроблено автором

У лістингу 3.5 наведено фрагмент створення замовлення під час checkout. Frontend викликає API-метод створення замовлення на основі поточного кошика. Після успішного виконання операції користувач переходить до сторінки замовлень. На backend-рівні створюються записи замовлення, позицій замовлення та оплати, що дозволяє надалі контролювати стан виконання покупки.

Лістинг 3.5 – Створення замовлення під час checkout

```
async function handlePlaceOrder() {
  try {
    setSubmitting(true);
    setMessage("");
```

```

        const order = await createOrderFromCart(paymentMethod);
        setMessage(`Order #${order.id} created successfully.`);
        navigate(path("/orders"));
    } catch (submitError) {
        setMessage(submitError.message);
    } finally {
        setSubmitting(false);
    }
}

export async function createOrderFromCart(provider = "card") {
    const response = await jsonRequest("/api/me/orders/from-cart", {
        method: "POST",
        body: JSON.stringify({ provider }),
    });
    return readJson(response);
}

```

кінець лістингу 3.5

На рисунку 3.6 показано панель продавця. У цій частині застосунку продавець може переглядати власні товари, додавати нові позиції, редагувати наявні товари, працювати з категоріями та переглядати замовлення, які стосуються його товарів. Наявність seller-панелі підтверджує, що система реалізує базову логіку маркетплейсу з розподілом ролей між покупцем і продавцем. Форма створення відгуку доступна покупцю не завжди. У системі передбачено обмеження, за яким користувач може залишити відгук лише після підтвердження оплати замовлення, у якому був відповідний товар.

Це обмеження підвищує достовірність відгуків, оскільки коментар може залишити лише користувач, який справді оформив і оплатив замовлення. Після створення відгуку його текст передається до AI-модуля, який визначає загальну тональність коментаря. Результат аналізу зберігається разом із відгуком і надалі може використовуватися для швидкого оцінювання реакції покупців на товар. Така функція не замінює ручну перевірку коментарів, але спрощує первинне визначення їх загального змісту. Завдяки цьому продавець отримує додаткову інформацію про сприйняття товару покупцями. Це підкреслює практичне використання AI-модуля в межах реалізованого маркетплейсу.

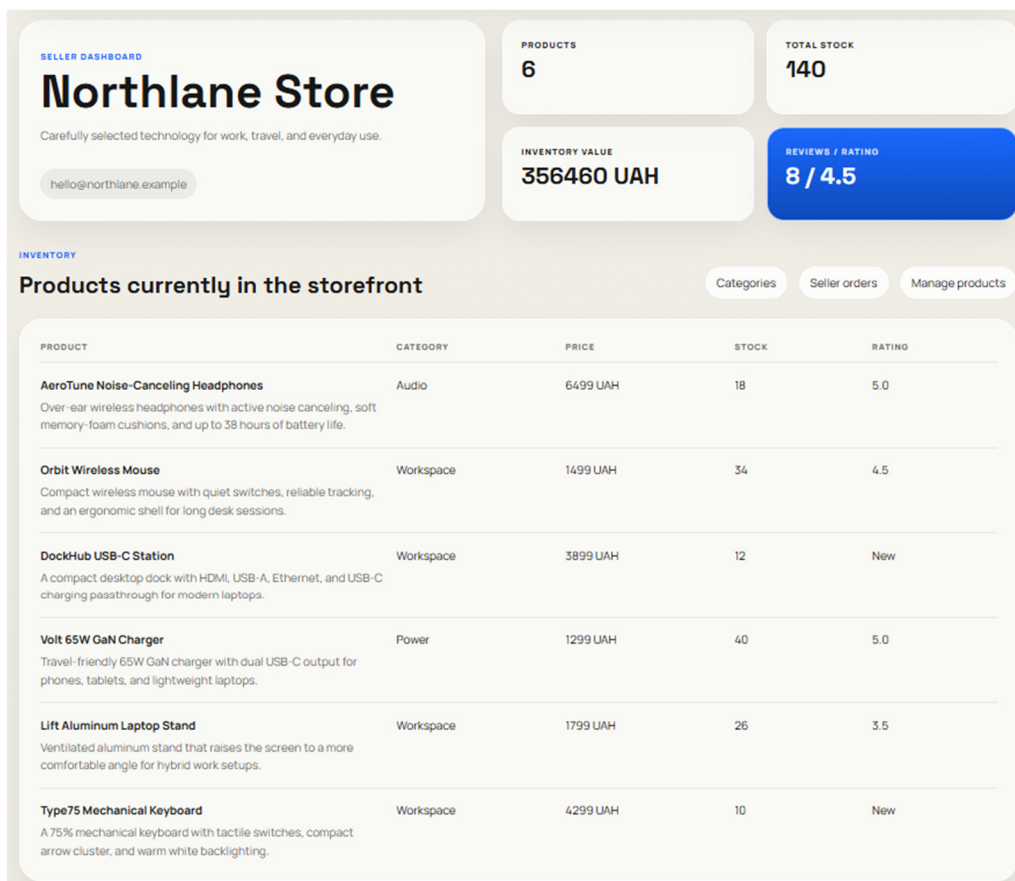


Рисунок 3.6 – Seller-панель керування товарами та замовленнями

Джерело: розроблено автором

У лістингу 3.6 показано backend-логіку підтвердження оплати замовлення продавцем. Спочатку перевіряється, чи користувач авторизований і чи має він роль продавця. Далі система перевіряє, чи належить замовлення цьому продавцю через товари, які входять до замовлення. Після успішної перевірки статус оплати набуває значення `paid`, а замовлення в межах прототипу позначається як доставлене. Такий механізм запобігає зміні чужих замовлень і забезпечує контроль доступу до seller-функцій.

Лістинг 3.6 – Підтвердження оплати замовлення продавцем

```
def update_current_seller_order_payment(order_id: int):
    payload = request.get_json(silent=True) or {}
    next_status = (payload.get("status") or "").strip().lower()

    if next_status not in {"paid"}:
        return jsonify({"error": "Only transition to paid is supported"}), 400
```

```

with session_scope() as session_db:
    user = get_session_user(session_db)
    if user is None:
        return jsonify({"error": "Authentication required"}),
401
    if user.role is None or user.role.name != "seller":
        return jsonify({"error": "Seller access required"}),
403

    order = session_db.scalar(
select(Order).join(Order.items).join(OrderItem.product)
        .where(Order.id == order_id, Product.seller_id ==
user.id)
        .options(selectinload(Order.payment))
    )
    if order.payment.status == "paid":
        return jsonify({"error": "Payment is already marked as
paid"}), 409

    order.payment.status = "paid"
    order.status = "delivered"
    session_db.flush()
    return jsonify(serialize_seller_order(order, user.id))

```

кінець лістингу 3.6

На рисунку 3.7 наведено форму створення відгуку. Вона відображається для покупця після підтвердження оплати замовлення. Користувач вводить текст коментаря та надсилає його до системи. Після цього backend зберігає відгук і передає його текст до AI-модуля для аналізу тональності.

Рисунок 3.7 – Форма створення відгуку після оплаченого замовлення

Джерело: розроблено автором

Результат AI-аналізу є додатковою функцією системи, яка дозволяє автоматично визначати загальний характер текстового відгуку. У межах

реалізованого застосунку результат класифікації може мати значення positive або negative.

У лістингу 3.7 наведено перевірку права покупця на створення відгуку. Backend перевіряє роль користувача, відсутність попереднього відгуку для цього товару та наявність оплаченої покупки.

Лістинг 3.7 – Перевірка права покупця на створення відгуку

```
def serialize_review_policy(session_db, product, user) -> dict:
    if user is None or user.role is None or user.role.name !=
"buyer":
        return {"can_submit": False, "message": "Only buyer
accounts can leave reviews."}
    existing_review = next(
        (review for review in product.reviews if review.user_id ==
user.id),
        None,
    )
    if existing_review is not None:
        return {"can_submit": False, "already_reviewed": True}
    has_paid_purchase = session_db.scalar(
        select(Order.id)
        .join(Order.items)
        .join(Order.payment)
        .where(
            Order.user_id == user.id,
            OrderItem.product_id == product.id,
            Payment.status == "paid",
        )
        .limit(1)
    ) is not None
    if not has_paid_purchase:
        return {"can_submit": False, "has_paid_purchase": False}
    return {"can_submit": True, "has_paid_purchase": True}
```

кінець лістингу 3.7

Така перевірка виконується саме на серверному рівні, тому користувач не може обійти це обмеження лише зміною інтерфейсу у браузері.

На рисунку 3.8 показано результат AI-аналізу відгуку. Після створення коментаря система передає його текст до AI-модуля, отримує результат класифікації тональності та зберігає цей результат разом із відгуком. Це дає змогу продавцю або адміністратору швидше оцінювати загальний характер реакцій покупців на товари.

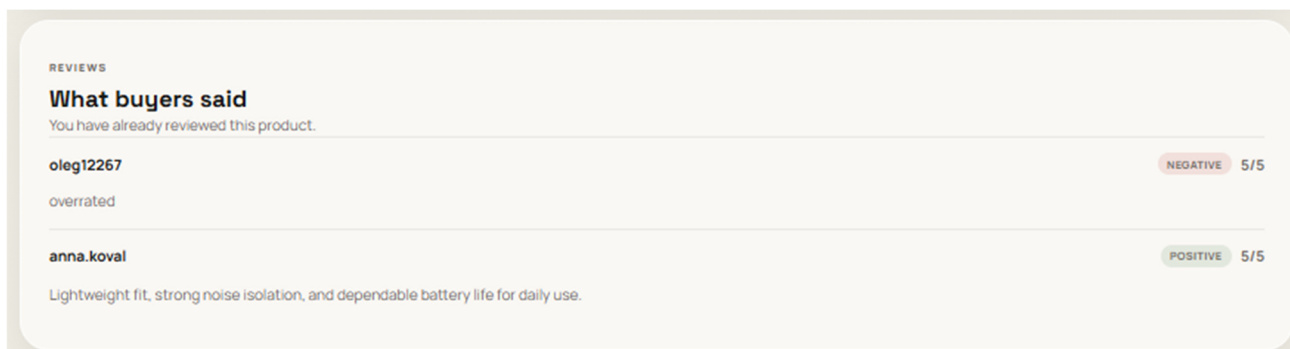


Рисунок 3.8 – Результат AI-аналізу відгуку

Джерело: розроблено автором

У лістингу 3.8 показано фрагмент AI-аналізу тональності текстового відгуку. Код нормалізує текст, формує prompt для моделі, викликає SageMaker endpoint через runtime API та очікує відповідь у вигляді одного з двох значень: positive або negative [15].

Лістинг 3.8 – AI-аналіз тональності текстового відгуку

```
def analyze_review_sentiment(text: str) -> str:
    normalized_text = " ".join(text.strip().split())
    if len(normalized_text) < 6:
        raise ValueError("Review text must contain at least 6
characters.")
    prompt = (
        "Classify the sentiment of this ecommerce product
review.\n"
        "Reply with one word only: positive or negative.\n\n"
        f"Review: {normalized_text}\n"
        "Sentiment:"
    )
    try:
        raw_text = _invoke_sagemaker_prompt(prompt,
max_new_tokens=8).lower()
        sentiment_match = re.search(r"\b(positive|negative)\b",
raw_text)
        if sentiment_match:
            return sentiment_match.group(1)
        except (ClientError, BotoCoreError, RuntimeError,
json.JSONDecodeError) as exc:
            app.logger.warning("SageMaker review sentiment failed:
%s", exc)
    return _heuristic_review_sentiment(normalized_text)
```

кінець лістингу 3.8

Якщо під час звернення до SageMaker виникає помилка або відповідь моделі не містить очікуваного результату, система не перериває роботу повністю, а використовує резервний евристичний механізм аналізу. Це підвищує стійкість застосунку та дозволяє зберегти відгук навіть у разі тимчасової недоступності AI-сервісу.

Для взаємодії frontend-частини з backend API використовується окремий API-клієнт. Він централізує формування URL, обробку JSON-відповідей, роботу з помилками та передавання cookies для сесійної автентифікації.

У лістингу 3.9 наведено API-клієнт frontend-частини. Він використовується в різних компонентах застосунку під час завантаження товарів, роботи з кошиком, створення замовлень, отримання відгуків і взаємодії із seller-функціями.

Лістинг 3.9 – API-клієнт frontend для взаємодії з backend

```
const API_BASE_URL = (import.meta.env.VITE_API_BASE_URL ||
  "").replace(/\/$/, "");
function buildUrl(path) {
  return API_BASE_URL ? `${API_BASE_URL}${path}` : path;
}
async function readJson(response) {
  if (!response.ok) {
    let message = "Request failed";
    try {
      const payload = await response.json();
      message = payload.error || message;
    } catch {
      message = response.statusText || message;
    }
    throw new Error(message);
  }
  return response.json();
}
function jsonRequest(url, options = {}) {
  const nextOptions = { ...options, credentials: "include" };
  if (nextOptions.body && !nextOptions.headers) {
    nextOptions.headers = { "Content-Type": "application/json" };
  }
  return fetch(buildUrl(url), nextOptions);
}
```

кінець лістингу 3.9

Централізація запитів спрощує підтримку frontend-коду та забезпечує однакову обробку помилок у різних частинах системи.

3.2 Тестування та налагодження системи інтернет-маркетплейсу

Після реалізації основних модулів було проведено тестування системи для трьох основних ролей: гостя, покупця та продавця. Перевірка охоплювала відкриття головної сторінки, перегляд каталогу, відкриття сторінки товару, додавання товару до кошика, оформлення замовлення, перегляд деталей замовлення, роботу seller-панелі, підтвердження оплати, створення відгуку та AI-аналіз тональності текстового коментаря.

Тестування проводилося за кількома напрямками: функціональна перевірка основних сценаріїв користувачів, перевірка HTTP-запитів у браузері, аналіз структури JSON-відповідей backend API, перевірка обробки помилок на клієнтській частині та тестування AI-модуля аналізу відгуків.

Результати функціонального тестування підтвердили, що основні сценарії роботи системи виконуються коректно. Окрему увагу було приділено перевірці обмежень доступу: гість не може оформити замовлення або залишити відгук, покупець може працювати лише зі своїми замовленнями, а продавець має доступ тільки до замовлень, які містять його товари. Узагальнені результати функціонального тестування основних сценаріїв роботи системи наведено в таблиці 3.1. Такий формат подання дозволяє коротко показати перевірені дії, очікуваний результат і фактичний стан виконання. Під час перевірки також враховувалася відповідність фактичної поведінки системи визначеним функціональним вимогам. Для кожного сценарію перевірявся не лише успішний результат, а й реакція системи на обмежені або недоступні дії. Це дозволило переконатися, що основна бізнес-логіка виконується на серверному рівні, а не лише приховується в інтерфейсі.

Таблиця 3.1 – Результати функціонального тестування інтернет-маркетплейсу

Сценарій тестування	Очікуваний результат	Фактичний результат
Відкриття головної сторінки гостем	Головна сторінка відкривається без авторизації	Сторінка відкривається коректно
Перегляд каталогу товарів	Користувач бачить список товарів	Каталог завантажується з backend API
Відкриття сторінки товару	Відображається детальна інформація про товар	Дані товару та відгуки відображаються коректно
Спроба додати товар до кошика гостем	Система спрямовує користувача до авторизації	Користувач переходить на сторінку входу
Додавання товару до кошика покупцем	Товар додається до кошика	Товар успішно зберігається в кошику
Оформлення замовлення	Створюється замовлення на основі кошика	Замовлення створюється і відображається у списку
Підтвердження оплати продавцем	Статус оплати набуває значення paid	Оплата успішно підтверджується
Створення відгуку після оплати	Покупець отримує доступ до форми відгуку	Відгук зберігається у системі
АІ-аналіз відгуку	Система визначає positive або negative	Результат аналізу зберігається разом із відгуком
Обробка помилки API	Користувач бачить повідомлення про помилку	Повідомлення відображається в інтерфейсі

На рисунку 3.9 зображено сторінку деталей замовлення покупця. Вона підтверджує, що після оформлення покупки система зберігає інформацію про товари, їх кількість, загальну суму, статус замовлення та пов'язані товарні позиції. Така сторінка є важливою для контролю виконання замовлення з боку користувача.

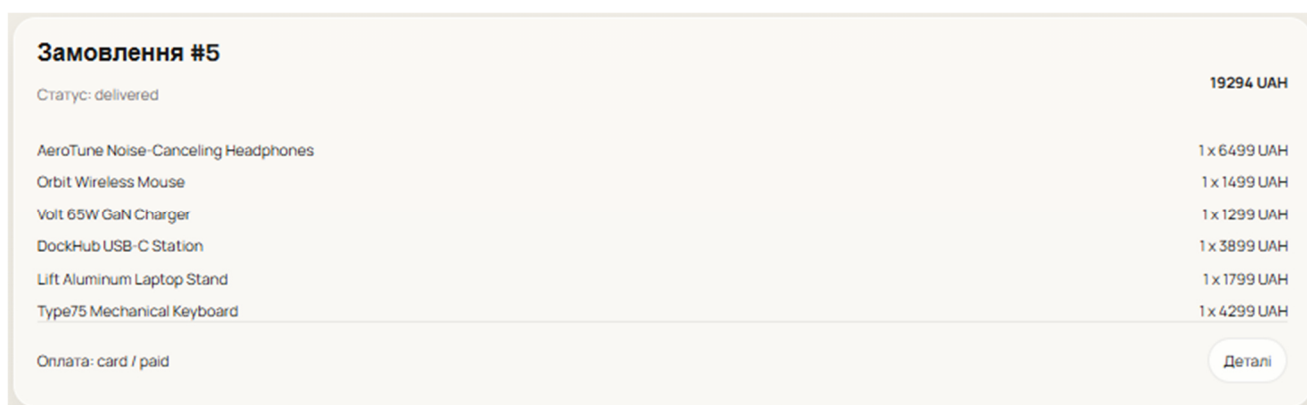


Рисунок 3.9 – Сторінка деталей замовлення покупця

Джерело: розроблено автором

Для перевірки взаємодії frontend-частини з backend API використовувалися інструменти розробника браузера. Вони дозволяють переглядати HTTP-запити, методи запитів, коди відповідей сервера, структуру переданих даних і JSON-відповіді, які повертаються з backend-частини.

У лістингу 3.10 наведено приклад відповіді API для товару. Backend повертає frontend-частині не лише основні характеристики товару, а й інформацію про категорію, продавця, відгуки та політику створення відгуку. Поле `review_policy` є важливим, оскільки воно визначає, чи може поточний користувач залишити відгук для цього товару.

Лістинг 3.10 – Приклад відповіді API для товару

```
{
  "id": 12,
  "title": "Wireless Headphones",
  "description": "Compact product card for the marketplace
catalog.",
  "price": 2499.0,
  "stock_qty": 14,
  "average_rating": 4.7,
  "category": { "id": 3, "name": "Audio" },
  "seller": { "id": 2, "shop_name": "Northlane Store" },
  "reviews": [
    {
      "id": 8,
      "rating": 5,
      "text": "Good quality and fast delivery.",
      "sentiment": "positive",
      "user_name": "Oleg"
    }
  ],
  "review_policy": {
    "can_submit": true,
    "has_paid_purchase": true,
    "already_reviewed": false
  }
}
```

кінець лістингу 3.10

У лістингу 3.11 показано обробку помилки API на клієнтській частині. Якщо під час завантаження даних виникає помилка, компонент зберігає повідомлення у стані `error` і відображає його користувачу. Завдяки цьому

інтерфейс не залишається порожнім і користувач отримує зрозуміле повідомлення про проблему.

Лістинг 3.11 – Обробка помилки API на клієнтській частині

```

async function loadData() {
  try {
    setLoading(true);
    setError("");
    const [productsData, categoriesData] = await Promise.all([
      fetchProducts(),
      fetchCategories(),
    ]);
    setProducts(productsData);
    setCategories(categoriesData);
  } catch (loadError) {
    setError(loadError.message);
  } finally {
    setLoading(false);
  }
}

if (error) {
  return <div className="state-card error">{error}</div>;
}

```

кінець лістингу 3.11

Окремо було проведено тестування AI-модуля аналізу відгуків, результати якого наведено в таблиці 3.2. Для цього було сформовано 100 тестових відгуків: 50 позитивних і 50 негативних. Тестові коментарі склалися з одного-двох речень і були сформульовані без очевидних маркерів тональності, щоб перевірити здатність моделі визначати зміст за контекстом, а не лише за окремими словами.

Таблиця 3.2 – Результати тестування AI-модуля аналізу відгуків

Тип відгуків	Кількість тестових відгуків	Правильно класифіковано	Точність
Позитивні	50	38	76 %
Негативні	50	46	92 %
Усього	100	84	78 %

Для перевірки роботи AI-модуля було сформовано тестову вибірку зі 100 відгуків, з яких 50 мали позитивну тональність і 50 – негативну. За результатами

тестування модель правильно класифікувала 84 відгуки зі 100, тому загальна точність становила 84 %. Для позитивних відгуків правильними були 38 результатів із 50, що відповідає 76 %. Для негативних відгуків модель правильно визначила 46 результатів із 50, що відповідає 92 %.

Отримані результати показали, що AI-модель краще визначає негативну тональність, тоді як частина позитивних контекстних формулювань може класифікуватися неправильно. Це є важливим результатом тестування, оскільки він демонструє як практичну придатність інтегрованого AI-модуля для автоматизованої оцінки відгуків, так і його обмеження під час аналізу складніших текстових формулювань.

За результатами тестування було визначено мінімальні системні вимоги для локального запуску програмного продукту, які наведені в таблиці 3.3. Для роботи backend-частини потрібні Python, встановлені залежності проєкту та доступ до файлу бази даних SQLite. Для роботи frontend-частини необхідні Node.js, npm і сучасний браузер.

Таблиця 3.3 – Мінімальні системні вимоги для запуску застосунку

Компонент	Мінімальна вимога
Операційна система	Windows 10/11 або Linux
Python	Версія 3.11 або новіша
Node.js	Версія 18 або новіша
Менеджер пакетів	npm
Браузер	Google Chrome, Microsoft Edge або Mozilla Firefox
Оперативна пам'ять	4 GB або більше
База даних	SQLite без окремого серверного процесу
Мережа	Доступ до інтернету для роботи з AWS SageMaker

Таким чином, тестування підтвердило працездатність основних функцій інтернет-маркетплейсу: перегляду товарів, роботи з кошиком, оформлення замовлень, seller-функцій, підтвердження оплати, створення відгуків і AI-аналізу тональності. Також було перевірено взаємодію frontend-частини з backend API, обробку помилок і коректність основних JSON-відповідей.

3.3 Посібник з інсталяції та використання програмного продукту

Для локального запуску програмного продукту необхідно мати встановлені Python, Node.js, npm та Git. Backend-частина розміщена у корені проєкту, а frontend-частина – у папці frontend. Після налаштування залежностей обидві частини можуть запускатися окремо або через підготовлений batch-скрипт.

Перед запуском backend-частини потрібно налаштувати змінні середовища. Вони визначають режим роботи застосунку, шлях до бази даних, адресу frontend-частини, параметри cookies і дані для підключення до AI-сервісу.

У лістингу 3.12 наведено приклад конфігурації змінних середовища backend. У ньому задаються параметри середовища розробки, секретний ключ, шлях до бази даних, адреса frontend та параметри підключення до AI-сервісу

Лістинг 3.12 – Приклад конфігурації змінних середовища backend

```
FLASK_ENV=development
SECRET_KEY=local-development-secret
DATABASE_URL=sqlite:///app.db
FRONTEND_URL=http://127.0.0.1:5173
AWS_REGION=us-east-1
SAGEMAKER_REGION=us-east-1
SAGEMAKER_ENDPOINT_NAME=final-project-flan-t5-base-endpoint
COOKIE_SECURE=false
COOKIE_SAMESITE=Lax
```

кінець лістингу 3.12

Значення SECRET_KEY=local-development-secret використовується лише як приклад для локальної розробки. У реальному розгортанні секретний ключ не повинен зберігатися у відкритому вигляді в репозиторії.

У лістингу 3.13 показано приклад команд для локального запуску backend та frontend. Спочатку виконується перехід до кореневої папки проєкту та запуск Flask-застосунку через Python-інтерпретатор із віртуального середовища. Далі виконується перехід до папки frontend, встановлення залежностей через npm install і запуск Vite development server на локальному порту 5173.

Лістинг 3.13 – Команди запуску backend та frontend

```
cd marketplace-project
.\.venv\Scripts\python.exe app.py

cd frontend
npm install
npm run dev -- --host 127.0.0.1 --port 5173

curl.exe http://127.0.0.1:5000/health
curl.exe -I http://127.0.0.1:5173/
```

кінець лістингу 3.13

Для спрощення локального запуску системи може використовуватися допоміжний batch-скрипт, який автоматично відкриває окремі процеси для Flask backend і Vite frontend.

На рисунку 3.10 показано головну сторінку інтернет-маркетплейсу

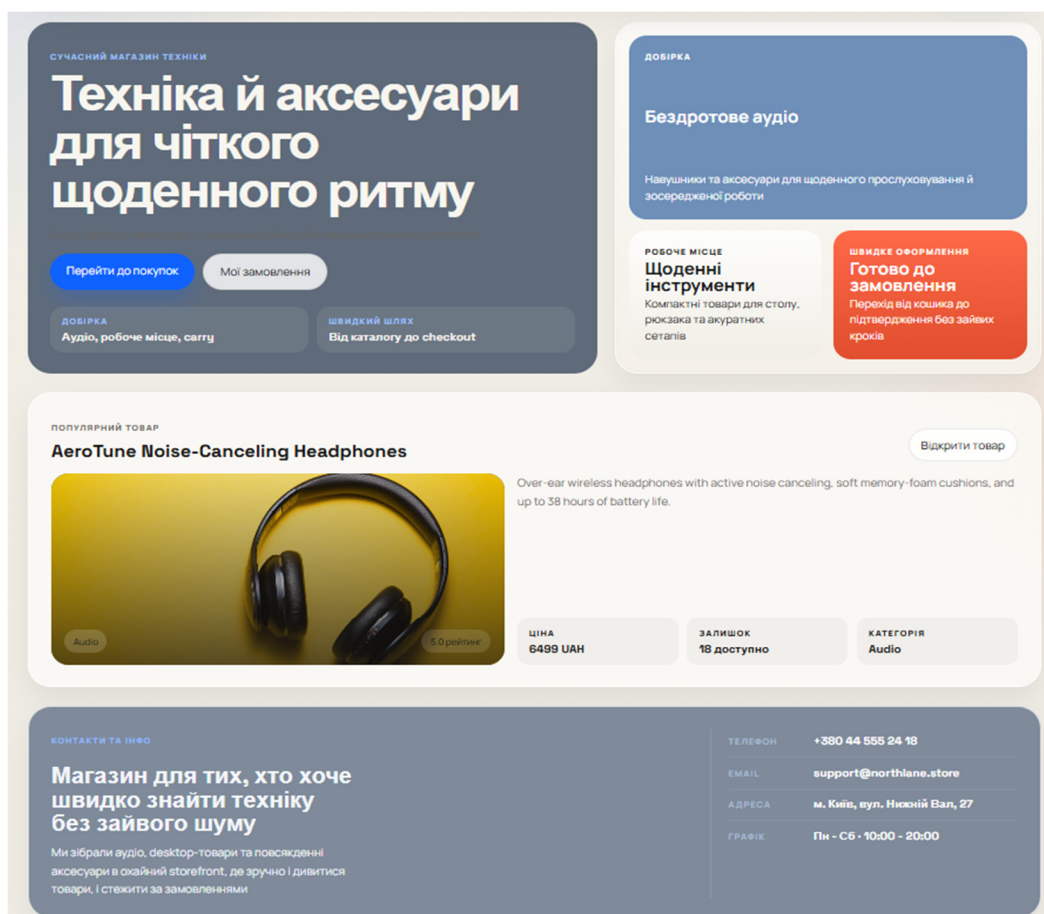


Рисунок 3.10 – Головна сторінка запущеного інтернет-маркетплейсу

Джерело: розроблено автором

Вона містить вступний інформаційний блок, навігацію, популярні товари та елементи переходу до каталогу. Відображення цієї сторінки підтверджує, що клієнтська частина застосунку запущена коректно.

Після запуску системи гість може переглядати відкриту частину сайту. До доступних для гостя сторінок належать головна сторінка, каталог товарів і сторінка окремого товару. Якщо неавторизований користувач намагається додати товар до кошика або залишити відгук, система спрямовує його до сторінки авторизації.

Покупець після входу в систему отримує доступ до кошика, checkout і сторінки власних замовлень. Для оформлення покупки користувач відкриває каталог, обирає товар, додає його до кошика, переходить до сторінки оформлення замовлення та підтверджує покупку. Після створення замовлення покупець може переглядати його деталі та поточний стан. Форма створення відгуку стає доступною лише після підтвердження оплати відповідного замовлення.

Продавець після авторизації отримує доступ до seller-панелі. У цій частині системи він може переглядати власні товари, створювати нові товарні позиції, редагувати наявні товари, працювати з категоріями та переглядати замовлення, які містять його товари. Також продавець може підтвердити оплату замовлення, після чого статус оплати набуває значення `paid`, а покупець отримує можливість залишити відгук.

Отже, третій розділ підтверджує практичну реалізацію основних функцій інтернет-маркетплейсу. У ньому описано frontend-частину, backend API, маршрутизацію, адаптивну навігацію, каталог, сторінку товару, кошик, оформлення замовлень, seller-панель, механізм створення відгуків, AI-аналіз тональності, тестування системи та порядок локального запуску програмного продукту.

РОЗДІЛ 4

СПЕЦІАЛЬНІ РОЗРАХУНКИ

4.1 Захист інформації в інтернет-маркетплейсі

Захист інформації є важливою складовою розробки інтернет-маркетплейсу, оскільки система працює з обліковими записами користувачів, ролями доступу, адресами доставки, замовленнями, товарами, платежами та відгуками. Навіть у межах навчально-практичного прототипу потрібно передбачити базові механізми безпеки, які зменшують ризик несанкціонованого доступу до seller-функцій, зміни чужих замовлень, створення недостовірних відгуків і отримання службових даних через API.

Основними об'єктами захисту в розробленій системі є облікові записи користувачів, персональні дані покупців, інформація про замовлення, дані товарів продавця, статуси оплати та результати AI-аналізу відгуків. Для цих об'єктів важливо забезпечити конфіденційність, цілісність і контрольований доступ. Під час проєктування захисних механізмів враховано загальні рекомендації щодо безпеки вебзастосунків [16]. Окремо враховано рекомендації щодо автентифікації користувачів [17].

У системі використовується рольова модель доступу. Передбачено три основні стани користувача: гість, покупець і продавець. Гість може переглядати відкриті сторінки, каталог і товари, але не може оформлювати замовлення або залишати відгуки. Покупець отримує доступ до кошика, checkout, власних замовлень і форми відгуку після підтвердження оплати. Продавець має доступ до seller-панелі, товарів, категорій і замовлень, які містять його товарні позиції.

Автентифікація користувача виконується через backend-частину. Під час входу користувач передає email і пароль, після чого сервер перевіряє їх відповідність даним, збереженим у базі. Якщо облікові дані правильні, у серверній сесії зберігається ідентифікатор користувача. Надалі frontend надсилає запити з cookies, а backend визначає поточного користувача, його роль і право на виконання конкретної дії.

Одним із базових елементів захисту є хешування паролів перед збереженням у базі даних. Пароль не повинен зберігатися у відкритому вигляді, оскільки у разі компрометації бази даних це створює прямий ризик для користувачів. Для цього застосовується одностороннє криптографічне перетворення, у результаті якого початковий пароль замінюється хеш-значенням. Таке значення використовується для подальшої перевірки пароля під час входу користувача в систему, але не дозволяє напряму відновити початковий пароль.

Для підвищення стійкості хешування до атак доцільно використовувати додаткове випадкове значення `salt` і параметр складності обчислень. Значення `salt` унеможливорює отримання однакових хешів для однакових паролів різних користувачів, а параметр складності збільшує обчислювальні витрати на підбір пароля. Завдяки цьому навіть у разі витоку бази даних злоумисник не отримує доступу до початкових паролів користувачів у відкритому вигляді.

Використання `salt` є важливим для захисту від попередньо обчислених таблиць хешів. Навіть якщо два користувачі мають однакові паролі, різні значення `salt` призводять до різних результатів хешування. Параметр складності дає змогу збільшити кількість обчислень, необхідних для формування хешу, і таким чином ускладнює атаки повного перебору. Авторизація у системі базується не лише на факті входу користувача, а й на перевірці його ролі та належності ресурсу. Наприклад, `seller`-сторінки доступні тільки користувачу з роллю `seller`. Крім того, продавець може переглядати та змінювати лише ті замовлення, які містять товари саме цього продавця. Такий підхід запобігає ситуації, коли один продавець отримує доступ до замовлень або товарів іншого продавця.

Для обмеження доступу до створення відгуків використовується окреме бізнес-правило. Покупець може залишити відгук лише тоді, коли він авторизований, має роль `buyer`, ще не залишав відгук для цього товару та має оплачене замовлення, у якому присутній відповідний товар. У цьому випадку перевіряється не статус замовлення, а стан пов'язаного запису `Payment`:

можливість створення відгуку відкривається після того, як статус оплати набуває значення paid.

Для узагальнення основних ризиків безпеки та способів їх зменшення сформовано таблицю 4.1.

Таблиця 4.1 – Основні ризики безпеки та способи їх зменшення

Ризик	Можливий наслідок	Механізм зменшення ризику
Збереження паролів у відкритому вигляді	Компрометація облікових записів у разі витоку бази даних	Хешування паролів із використанням salt і параметра складності
Доступ покупця до seller-функцій	Несанкціонована зміна товарів або замовлень	Перевірка ролі користувача на backend-рівні
Зміна чужого замовлення продавцем	Порушення цілісності даних і прав інших продавців	Перевірка належності товарів у замовленні конкретному продавцю
Створення недостовірних відгуків	Спотворення оцінки товарів і довіри покупців	Дозвіл на відгук лише після підтвердженої оплаченої покупки
Прямий доступ frontend до бази даних	Неконтрольована зміна або читання даних	Доступ до SQLite тільки через backend API
Недоступність AI-сервісу	Неможливість класифікації тональності відгуку	Обробка винятків і резервний механізм аналізу

Захист взаємодії між клієнтською та серверною частинами забезпечується через контроль HTTP-запитів. Backend повертає структуровані відповіді у форматі JSON і використовує коди стану HTTP для повідомлення про результат виконання операцій. Наприклад, у разі відсутності авторизації повертається код 401, у разі нестачі прав – 403, а у разі некоректних даних – 400. Це дозволяє frontend-частині коректно обробляти помилки й не відкривати користувачу дії, які йому недоступні [18].

Окрему роль відіграє налаштування CORS. Оскільки frontend і backend під час розробки можуть працювати на різних портах, серверна частина повинна явно дозволяти запити з довіреної адреси клієнтського застосунку. CORS не замінює автентифікацію та авторизацію, однак допомагає контролювати браузерну взаємодію між клієнтською та серверною частинами [19].

Під час розгортання застосунку у хмарному середовищі важливо використовувати HTTPS. Захищене з'єднання через TLS зменшує ризик перехоплення або модифікації даних під час передавання між браузером

користувача і сервером. Для маркетплейсу це важливо, оскільки через мережу передаються дані входу, інформація про замовлення, адреси доставки та дії продавця.

База даних системи містить взаємопов'язані сутності користувачів, товарів, кошиків, замовлень, платежів і відгуків. Для зменшення ризику некоректної зміни даних усі операції виконуються через backend API, а не напряму з клієнтської частини. Frontend не має прямого доступу до SQLite-файлу, тому кожна операція проходить через перевірки серверної логіки.

AI-модуль аналізу відгуків також потребує контрольованої інтеграції. До моделі передається лише текст відгуку, необхідний для визначення тональності, а результат обробки зберігається як коротке значення `positive` або `negative`. Такий підхід зменшує обсяг даних, що передаються до зовнішнього сервісу, і дозволяє відокремити бізнес-логіку маркетплейсу від механізму машинного аналізу.

Для підвищення стійкості системи backend повинен передбачати обробку винятків під час звернення до AI-сервісу. Якщо модель тимчасово недоступна або повертає некоректну відповідь, система не повинна втрачати відгук користувача чи повністю переривати роботу сторінки. У такому випадку доцільно логувати помилку та використовувати контрольований резервний механізм аналізу.

Для зменшення ризику перевантаження API у майбутньому доцільно передбачити `rate limiting`. Обмеження кількості запитів з одного джерела допоможе захистити маршрути входу, реєстрації, створення замовлень і відгуків від надмірного навантаження або автоматизованих спроб зловживання. У хмарному середовищі подібні обмеження можуть бути реалізовані на рівні `reverse proxy`, API gateway або самого backend-застосунку.

Отже, спеціальні розрахунки та обґрунтування заходів безпеки для розробленого інтернет-маркетплейсу охоплюють захист облікових записів, хешування паролів, рольову авторизацію, контроль доступу до seller-функцій, перевірку права на створення відгуків, захист API-запитів, використання HTTPS, ізоляцію бази даних від клієнтської частини та інтеграцію AI-модуля.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було розроблено інтернет-маркетплейс на основі Python. Система реалізована як навчально-практичний MVP-прототип і забезпечує основні процеси електронної комерції: перегляд товарів, роботу з кошиком, оформлення замовлень, seller-панель, створення відгуків та AI-аналіз їх тональності.

У роботі було проаналізовано предметну область електронної комерції та визначено особливості інтернет-маркетплейсів. На відміну від звичайного інтернет-магазину, маркетплейс передбачає взаємодію кількох груп користувачів, тому в системі реалізовано ролі гостя, покупця та продавця. Гість має доступ до відкритої частини сайту, покупець працює з кошиком, замовленнями та відгуками, а продавець керує товарами й замовленнями через seller-панель.

Було досліджено аналогічні програмні рішення та встановлено, що великі маркетплейси мають складну платіжну, логістичну й аналітичну інфраструктуру. Тому для реалізації було обрано напрям створення компактного прототипу, який відображає базову логіку роботи маркетплейсу та може бути розширений у майбутньому.

Обґрунтовано вибір засобів розробки системи. Клієнтську частину реалізовано з використанням React, Vite та React Router, серверну частину мовою Python із використанням Flask. Для збереження даних використано SQLite та SQLAlchemy, а для аналізу текстових відгуків інтегровано AI-модуль. Такий набір технологій дав змогу розділити frontend, backend, базу даних і модуль інтелектуальної обробки тексту.

У межах проєктування було сформульовано функціональні та нефункціональні вимоги, розроблено функціонально-структурну схему, UML-діаграми, блок-схеми алгоритмів і модель бази даних. Побудовані схеми відображають архітектуру системи, сценарії роботи гостя, покупця і продавця,

логіку авторизації, оформлення замовлення, створення відгуку та взаємодію з базою даних.

Практична реалізація охопила створення frontend- і backend-частин вебзастосунку. У системі реалізовано каталог товарів, сторінку окремого товару, кошик, checkout, сторінку замовлень, seller-панель, підтвердження оплати та створення відгуків після оплаченої покупки. AI-модуль визначає тональність текстового відгуку як positive або negative і зберігає результат разом із коментарем.

Проведене тестування підтвердило працездатність основних сценаріїв роботи системи для гостя, покупця та продавця. Було перевірено перегляд товарів, додавання до кошика, оформлення замовлення, підтвердження оплати, створення відгуку та роботу AI-модуля. Загальна точність класифікації відгуків становила 84 %, що підтверджує можливість використання AI-модуля як допоміжного інструмента для оцінки коментарів користувачів.

Також було розглянуто питання захисту інформації. У системі передбачено автентифікацію, рольову авторизацію, перевірку доступу до seller-функцій, обмеження створення відгуків, хешування паролів, налаштування CORS, використання HTTPS/TLS та відокремлення frontend-частини від прямого доступу до бази даних. Отже, поставлену мету роботи досягнуто, а розроблений інтернет-маркетплейс може бути використаний як основа для подальшого розвитку системи електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузьмич О. Р. (студент), Тулашвілі Ю. Й. Інтернет-маркетплейс на основі Python. Тези доповідей I Міжнародної науково-практичної конференції «Сучасні інформаційні технології: від теорії до практики (СІнТех-2026)» (22-23 травня 2026 року). Луцьк: ЛНТУ, 2026. С. 89-91.
2. Соколик А. А., Лежєп'юкова В. Г. Стан та перспективи розвитку електронної комерції в Україні. Міжнародні економічні відносини. Сталий розвиток України в умовах глобалізації та Європейської економічної інтеграції: проблеми, перспективи, ефективність. ФЕНІКС-2025 : матеріали міжнар. наук.-практ. конф., 28 травня 2025 р., м. Харків. Харків : НТУ «ХПІ», 2025. С. 112-116.
3. Левченко В. М., Жигалкевич Ж. М. Сучасні підходи до формування асортиментних стратегій на споживчому ринку. Матеріали V Міжнародної науково-практичної конференції «Бізнес, інновації, менеджмент: проблеми та перспективи», м. Київ, 25 квіт. 2024 р. Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2024. С. 77-78.
4. Тебенко В. М. Особливості е-комерції в Україні. Сучасні виклики та сталий розвиток економіки і бізнесу : матеріали I Всеукраїнської науково-практичної конференції, м. Запоріжжя, 10 грудня 2024 р. Запоріжжя : ТДАТУ, 2024. С. 319-322.
5. React Documentation. URL: <https://react.dev/> (дата звернення: 04.04.2026).
6. Vite Documentation. URL: <https://vite.dev/guide/> (дата звернення: 07.04.2026).
7. React Router Documentation. URL: <https://reactrouter.com/> (дата звернення: 10.04.2026).
8. Python Documentation. URL: <https://docs.python.org/3/> (дата звернення: 14.04.2026).
9. Flask Documentation. URL: <https://flask.palletsprojects.com/> (дата звернення: 17.04.2026).

10. SQLAlchemy Documentation. URL: <https://docs.sqlalchemy.org/> (дата звернення: 22.04.2026).
11. Amazon Web Services. Amazon SageMaker Documentation. URL: <https://docs.aws.amazon.com/sagemaker/> (дата звернення: 25.04.2026).
12. Amazon Web Services. Amazon Simple Storage Service Documentation. URL: <https://docs.aws.amazon.com/s3/> (дата звернення: 29.04.2026).
13. Amazon Web Services. Amazon CloudFront Documentation. URL: <https://docs.aws.amazon.com/cloudfront/> (дата звернення: 02.05.2026).
14. SQLite Documentation. URL: <https://www.sqlite.org/docs.html> (дата звернення: 06.05.2026).
15. Amazon Web Services. SageMaker Runtime API Reference. URL: https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_runtime_InvokeEndpoint.html (дата звернення: 09.05.2026).
16. OWASP Foundation. OWASP Top Ten. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 12.05.2026).
17. OWASP Foundation. Authentication Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 15.05.2026).
18. MDN Web Docs. HTTP response status codes. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status> (дата звернення: 17.05.2026).
19. MDN Web Docs. Cross-Origin Resource Sharing (CORS). URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> (дата звернення: 19.05.2026).

ДОДАТКИ

Додаток А

Сертифікат про участь у міжнародній науково-практичній конференції



CINTEX 2026

СЕРТИФІКАТ

засвідчує, що

Кузьмич Олег Ростиславович

взяв участь у

I Міжнародній науково-практичній конференції:
Сучасні інформаційні технології: від теорії до практики

загальним обсягом 15 годин (0,5 кредитів ECTS),
 яка проводилася **22-23 травня 2026 року**
 в Луцькому національному технічному університеті



[mintech-conf.lntu.edu.ua](http://mintech-conf.lntu.edu.ua/cert/2026-196)
 /cert/2026-196

Ректор ЛНТУ



Ірина ВАХОВИЧ

Луцький національний технічний університет
 вул. Львівська, 75 м. Луцьк
 Волинська обл. 43018

