Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ КЛІЄНТ-СЕРВЕРНОГО ВЕБЗАСТОСУНКУ E-COMMERCE З ВИКОРИСТАННЯМ СУЧАСНИХ ТЕХНОЛОГІЙ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-3
Об’єздова А. В.
(прізвище, ініціали)

Спеціальність 122 Комп’ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д. О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Об'єздовій Анастасії Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення клієнт-серверного вебзастосунку e-commerce з використанням сучасних вебтехнологій

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 22 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література з проєктування та розроблення вебзастосунків і систем електронної комерції, матеріали профільних конференцій, дані мережі Інтернет, документація та бібліотеки з відкритим кодом (Node.js, Express, Firebase Firestore, Amazon S3, Nodemailer), відкриті стандарти проєктування REST API та принципи чистого коду, матеріали щодо проєктування баз даних документоорієнтованого типу.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області та обґрунтування архітектури клієнт-серверного вебзастосунку електронної комерції.
2. Проєктування структури бази даних, клієнтської та серверної частин вебзастосунку.
3. Реалізація взаємодії клієнтської та серверної частин через REST API.
4. Тестування функціональних можливостей системи та аналіз результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) UML-діаграми проєктування системи (варіантів використання, діяльності), схема структури бази даних із зазначенням колекцій та зв'язків між ними, схема клієнт-серверної архітектури вебзастосунку, комп'ютерні ілюстрації роботи клієнтського інтерфейсу та сторінок управління товарами продавця.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-05.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Руденко Д. О._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 78 с., 1 табл., 29 рис., 37 джерело.

ВЕБЗАСТОСУНОК, ЕЛЕКТРОННА КОМЕРЦІЯ, AMAZON S3, EXPRESS, FIREBASE FIRESTORE, NODE.JS, REST API.

Об’єктом роботи є процес організації електронної комерції у форматі взаємодії між продавцями та покупцями через вебзастосунок.

Метою роботи є розроблення клієнт-серверного вебзастосунку електронної комерції, який забезпечує взаємодію між продавцями та покупцями, підтримує управління товарами, оформлення замовлень та роботу користувацького інтерфейсу.

Практична цінність полягає в автоматизації основних бізнес-процесів електронної комерції: перегляду каталогу, управління кошиком, оформлення замовлень та керування товарами продавця.

Розроблено вебзастосунок із реєстрацією та авторизацією користувачів, пошуком товарів, кошиком і списком бажаного, системою ролей, а також інтеграцією із зовнішніми сервісами для зберігання зображень і надсилання повідомлень.

Область застосування: онлайн-торгівля, маркетплейс-платформи, системи управління товарами та замовленнями.

AMAZON S3, E-COMMERCE, EXPRESS, FIREBASE FIRESTORE, NODE.JS, REST API, WEB APPLICATION.

Object of the work is the process of organizing e-commerce in the format of interaction between sellers and buyers through a web application.

Goal of the work is to develop a client-server e-commerce web application that enables interaction between sellers and buyers, supports product management, order processing, and user interface functionality.

Practical value lies in the automation of core e-commerce business processes: browsing product catalogues, managing shopping carts, placing orders, and providing sellers with tools to manage their products.

A web application has been developed that supports user registration and authentication, product search, shopping cart and wish list management, a user role system, and integration with external services for image storage and email notifications.

Application area: online retail, marketplace platforms, product and order management systems.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 9
1 Аналіз предметної області та постановка задачі 10
1.1 Аналіз предметної області електронної комерції 10
1.2 Аналіз існуючих вебзастосунків електронної комерції 12
1.3 Аналіз сучасних методів побудови вебзастосунків 18
1.3.1 Клієнт-серверна архітектура вебзастосунків 18
1.3.2 Фронтенд та бекенд вебзастосунків 19
1.3.3 REST API 21
1.4 Постановка задачі 22
2 Проєктування та моделювання вебзастосунку 24
2.1 Обґрунтування архітектури вебзастосунку 24
2.2 Моделювання функціональної структури системи 27
2.3 Проєктування бізнес-процесів вебзастосунку 29
2.4 Проєктування структури бази даних 33
2.5 Обґрунтування вибору технологій та сервісів 36
2.5.1 Обґрунтування вибору технологій клієнтської частини 36
2.5.2 Обґрунтування вибору технологій серверної частини 38
2.5.3 Обґрунтування вибору бази даних та зовнішніх сервісів 38
3 Реалізація та тестування вебзастосунку 40
3.1 Реалізація серверної частини вебзастосунку 42
3.1.1 Налаштування серверного середовища 42
3.1.2 Реалізація реєстрації та авторизації користувачів 43
3.1.3 Реалізація роботи з товарами 45
3.1.4 Реалізація оформлення замовлення 46
3.2 Реалізація клієнтської частини вебзастосунку 48
3.2.1 Реалізація основних сторінок вебзастосунку 48
3.2.2 Реалізація взаємодії клієнтської частини з API 50

3.3 Реалізація системи ролей та прав доступу 51
3.4 Реалізація роботи з базою даних 53
3.5 Інтеграція із зовнішніми сервісами 55
3.6 Демонстрація роботи вебзастосунку 58
3.6.1 Демонстрація роботи користувацької частини 58
3.6.2 Демонстрація роботи продавця в системі 66
3.7 Тестування вебзастосунку 70
3.8 Аналіз результатів та перспективи подальшого розвитку 71
Висновки 73
Перелік джерел посилання 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

API – Application Programming Interface (інтерфейс програмного застосунку, використовується для обміну даними між клієнтською та серверною частинами)

AWS – Amazon Web Services (хмарна платформа компанії Amazon)

B2C – Business-to-Consumer (модель електронної комерції «бізнес для споживача»)

bcrypt – бібліотека для хешування паролів перед їх збереженням у базі даних

CSS – Cascading Style Sheets (каскадні таблиці стилів, використовуються для візуального оформлення вебсторінок)

dotenv – інструмент для зберігання конфіденційних параметрів у змінних середовища окремо від програмного коду

e-commerce – електронна комерція, сукупність комерційних процесів, що здійснюються із використанням цифрових технологій та електронних засобів комунікації

fetch API – браузерний інтерфейс для виконання асинхронних HTTP-запитів до серверної частини

HTML – HyperText Markup Language (мова гіпертекстової розмітки, використовується для створення структури вебсторінок)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту, основа взаємодії між клієнтом і сервером)

JSON – JavaScript Object Notation (формат обміну структурованими даними між клієнтом і сервером)

LocalStorage – браузерне сховище для збереження даних на стороні клієнта між переходами між сторінками

REST – Representational State Transfer (архітектурний стиль побудови вебсервісів на основі HTTP-протоколу)

S3 – Simple Storage Service (хмарне сховище Amazon для збереження медіафайлів)

UML – Unified Modeling Language (уніфікована мова моделювання для опису структури та поведінки систем)

URL – Uniform Resource Locator (уніфікований локатор ресурсу, адреса файлу або сторінки в мережі)

UX/UI – User Experience / User Interface (досвід користувача / інтерфейс користувача)

ВСТУП

Зростання обсягів інтернет-торгівлі та активна цифровізація бізнес-процесів формують стійкий попит на сучасні програмні рішення у сфері електронної комерції. В Україні цей процес також набирає обертів: кількість онлайн-магазинів та маркетплейс-платформ щороку зростає, що підвищує конкуренцію та вимоги до якості вебзастосунків. Сучасний покупець очікує від онлайн-платформи не лише широкого асортименту товарів, а й інтуїтивно зрозумілого інтерфейсу, швидкої навігації та зручного процесу оформлення замовлення.

Аналіз існуючих платформ електронної комерції виявив низку типових недоліків: перевантаженість інтерфейсу рекламними блоками, складну структуру навігації, надмірно розтягнуті форми взаємодії та відсутність фіксованого навігаційного меню під час прокручування сторінки. Зазначені недоліки безпосередньо впливають на зручність роботи користувача з платформою та знижують ефективність виконання основних операцій. Це обґрунтовує необхідність розроблення нового вебзастосунку електронної комерції з покращеним користувацьким досвідом.

Для вирішення виявлених проблем у роботі застосовано клієнт-серверну архітектуру з чітким розподілом відповідальності між фронтенд та бекенд-компонентами, а взаємодію між частинами системи організовано через REST API. Серверну частину реалізовано на основі Node.js та Express, для зберігання даних використано Firebase Firestore, а медіафайли розміщено у хмарному сховищі Amazon S3.

Актуальність роботи полягає у зростанні попиту на сучасні «e-commerce» вебзастосунки із зрозумілим інтерфейсом, адаптивним дизайном та ефективною клієнт-серверною взаємодією. Результати роботи можуть бути використані малим та середнім бізнесом для розгортання власних онлайн-магазинів, а розроблена архітектура може бути використана як основа для подальшого масштабування та інтеграції додаткових модулів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області електронної комерції

Стрімкий розвиток інформаційних технологій та глобальної мережі Інтернет суттєво змінив підходи до ведення бізнесу, способи взаємодії між продавцями й покупцями та механізми здійснення торговельних операцій. Цифровізація суспільства сприяла появі нових моделей комерційної діяльності, у яких більшість бізнес-процесів здійснюється за допомогою онлайн-технологій. Одним із ключових напрямів цифрової економіки стала електронна комерція, яка забезпечує продаж товарів і послуг із використанням вебтехнологій, цифрових платформ та електронних платіжних систем [1]. З моменту свого зародження у 1995 році вона трансформувалась із інструменту онлайн-роздрібної торгівлі на багатофункціональну платформу, що кардинально змінює принципи функціонування ринків, бізнес-моделі підприємств і поведінку споживачів у глобальному масштабі [2].

Під електронною комерцією розуміють сукупність комерційних процесів, що здійснюються із використанням цифрових технологій та електронних засобів комунікації, зокрема купівлю та продаж товарів, електронні платежі, обмін інформацією між учасниками бізнесу, онлайн-банкінг та цифровий маркетинг [1]. Вона є складовою електронного бізнесу та передбачає виконання фінансових і торговельних операцій без фізичної присутності сторін [3]. Найбільш поширеною моделлю є B2C, у межах якої продаж продукції здійснюється безпосередньо кінцевому споживачу через вебмагазин або іншу онлайн-платформу [1].

Сучасні вебмагазини являють собою складні вебзастосунки, що поєднують каталог товарів, систему пошуку, кошик покупця, модулі оформлення замовлення, електронні платіжні сервіси та механізми доставки. Використання сучасних цифрових технологій забезпечує електронній

комерції можливість значно спростити процес придбання товарів та підвищити швидкість взаємодії між користувачем і продавцем [4]. Крім того, онлайн-торгівля забезпечує можливість автоматизації значної частини бізнес-процесів, включаючи обробку замовлень, управління складом, аналітику продажів та комунікацію з клієнтами [4].

Популярність вебмагазинів зумовлена активним розвитком цифрової економіки та зміною поведінки сучасних користувачів. Покупці дедалі частіше надають перевагу онлайн-покупкам через зручність, економію часу та можливість отримати доступ до широкого асортименту товарів незалежно від географічного розташування [4]. Вебмагазини функціонують цілодобово та забезпечують доступ до послуг із будь-якого пристрою, підключеного до мережі інтернет [3]. Для споживачів перевагами електронної торгівлі є швидкий пошук товарів, порівняння цін, перегляд характеристик продукції, перегляду відгуків інших покупців і використання персоналізованих рекомендацій [4].

Для бізнесу електронна комерція відкриває можливість розширення аудиторії клієнтів, зменшення витрат на утримання фізичних торговельних точок та підвищення ефективності управління комерційними процесами [4]. Онлайн-платформи дозволяють автоматизувати оплату, доставку та обробку замовлень, а також інтегрувати сучасні маркетингові інструменти, аналітичні системи та сервіси взаємодії із користувачами [1]. Водночас розвиток електронної торгівлі стимулює появу нових робочих місць у сферах IT-розробки, цифрового маркетингу, логістики та підтримки клієнтів [4].

Однією з головних тенденцій розвитку електронної комерції є стрімке зростання обсягів онлайн-торгівлі: частка онлайн-транзакцій та кількість споживачів, які купують через інтернет, щорічно збільшується [4]. Розвиток цифрових платіжних систем, логістичних сервісів та соціальних мереж додатково стимулює розширення цього ринку [4]. Загалом інформаційні технології відіграють важливу роль у підвищенні ефективності прийняття

рішень та оцінювання результатів діяльності в різних галузях, зокрема й у сфері електронної комерції [5].

Важливу роль у сучасних «e-commerce» вебзастосунках відіграють UX/UI-технології. UX визначає зручність взаємодії користувача із системою та спрямований на створення логічної структури вебзастосунку, що дозволяє швидко виконувати потрібні дії, зокрема через зрозумілу навігацію та мінімізацію когнітивного навантаження [6]. Навігація прямо впливає на конверсію: якщо користувач не може знайти потрібний товар, він не зможе його придбати, а навіть незначні навігаційні недоліки здатні спонукати його залишити сайт [7]. UI, своєю чергою, відповідає за візуальне оформлення інтерфейсу: кольорову гаму, типографіку, кнопки, форми та інші графічні елементи [6].

Попри переваги електронної комерції, існують проблеми захисту персональних даних, ризиків шахрайства, безпеки транзакцій та високої конкуренції між онлайн-магазинами [1]. Тому під час розроблення сучасних електронної комерції важливо забезпечити стабільну роботу системи, зрозумілий інтерфейс, безпечну обробку даних та оптимізацію функціональних можливостей для зручної роботи користувача. Сучасні технології підтримки прийняття рішень в інформаційних системах також спрямовані на підвищення надійності та ефективності функціонування таких систем [8].

1.2 Аналіз існуючих вебзастосунків електронної комерції

Для аналізу сучасних вебзастосунків електронної комерції було обрано платформи Amazon, OLX та Prom, оскільки вони реалізують різні моделі електронної комерції та охоплюють широкий спектр функцій взаємодії між продавцями і покупцями. Amazon є міжнародною маркетплейс-платформою з розвиненою системою пошуку, рекомендацій та оформлення замовлень, OLX

представляє модель онлайн-дошки оголошень із прямою взаємодією між користувачами, а Prom поєднує функціональність маркетплейсу з інструментами для створення інтернет-магазинів. Аналіз цих платформ дозволяє дослідити сучасні підходи до реалізації інтерфейсу та виявити основні недоліки існуючих рішень.

Amazon є однією з найбільших маркетплейс-платформ електронної комерції у світі, що забезпечує взаємодію між продавцями і покупцями, продаж товарів різних категорій, онлайн-оплату, оформлення замовлень, персоналізовані рекомендації та кабінети продавців. Загальний вигляд головної сторінки Amazon наведено на рисунку 1.1.

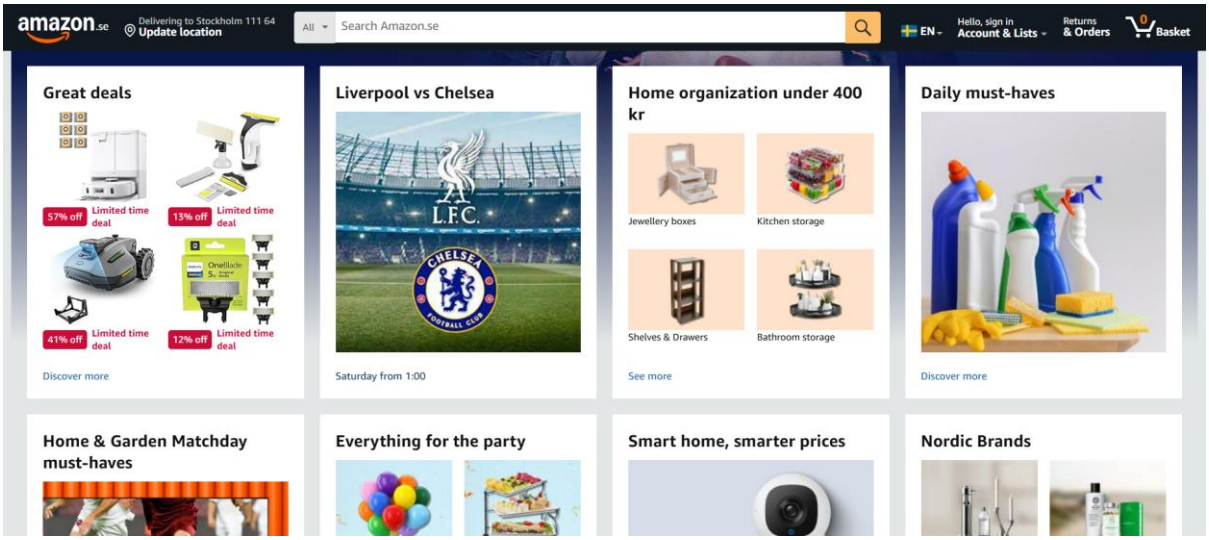


Рисунок 1.1 – Інтерфейс головної сторінки вебзастосунку Amazon

Amazon має широкий функціонал, який забезпечує зручну взаємодію користувачів із платформою. До основних переваг вебзастосунку належать ефективна система пошуку та фільтрації товарів, підтримка великої кількості продавців, адаптивний дизайн і наявність системи персоналізованих рекомендацій. Крім того, платформа реалізує зручний процес оформлення замовлень, що спрощує виконання покупок для користувачів.

Приклад реалізації каталогу товарів та системи фільтрації показано на рисунку 1.2.

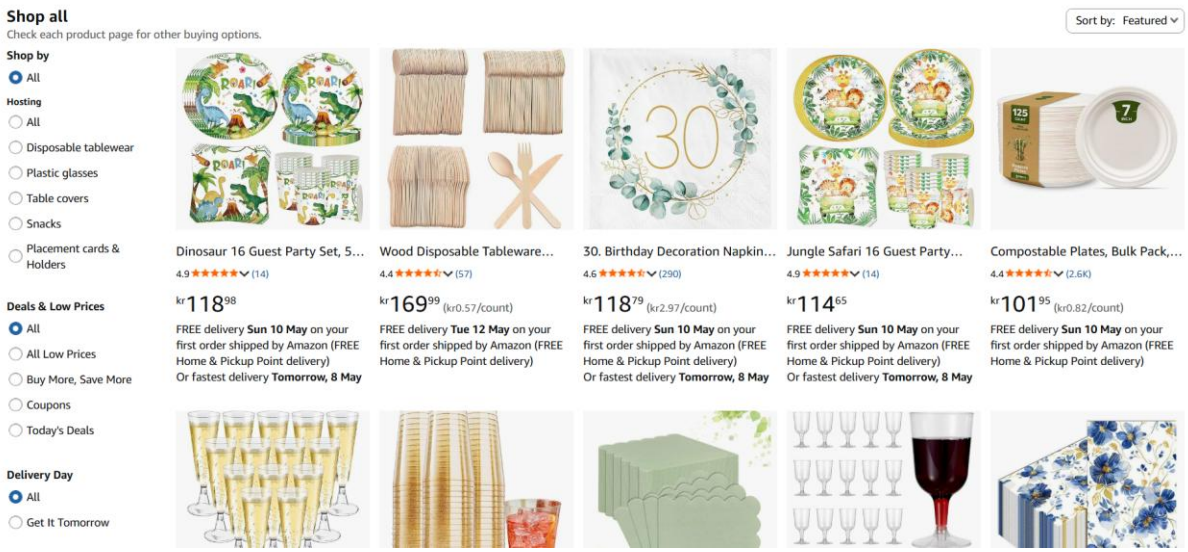


Рисунок 1.2 – Інтерфейс каталогу товарів та системи фільтрації Amazon

Разом із цим Amazon має й певні недоліки: інтерфейс перевантажений елементами та інформаційними блоками, структура навігації є складною для нових користувачів, а значна кількість рекламних і рекомендаційних блоків ускладнює сприйняття основного контенту. Архітектура платформи є складною та потребує значних ресурсів для реалізації, що робить подібний підхід не завжди доцільним для невеликих вебзастосунків електронної комерції. Окремим недоліком є відсутність фіксованого навігаційного меню під час прокручування сторінки: переглядаючи товари, користувач змушений повертатися у верхню частину сторінки для переходу до кошика, пошуку чи інших розділів, що ускладнює навігацію та збільшує час взаємодії з платформою.

OLX є платформою електронної комерції, яка працює за моделлю онлайн-дошки оголошень та забезпечує взаємодію між продавцями та покупцями. Основне призначення сервісу полягає у публікації оголошень про продаж товарів і послуг у різних категоріях. Платформа підтримує як продаж нових товарів, так і реалізацію вживаної продукції між приватними особами. Загальний вигляд головної сторінки OLX наведено на рисунку 1.3.

Основними функціональними можливостями OLX є створення та редагування оголошень, система пошуку товарів, категорії та фільтрація,

обмін повідомленнями між користувачами, а також можливість додавання товарів до обраного. Крім того, платформа підтримує авторизацію користувачів, особистий кабінет та систему сповіщень.

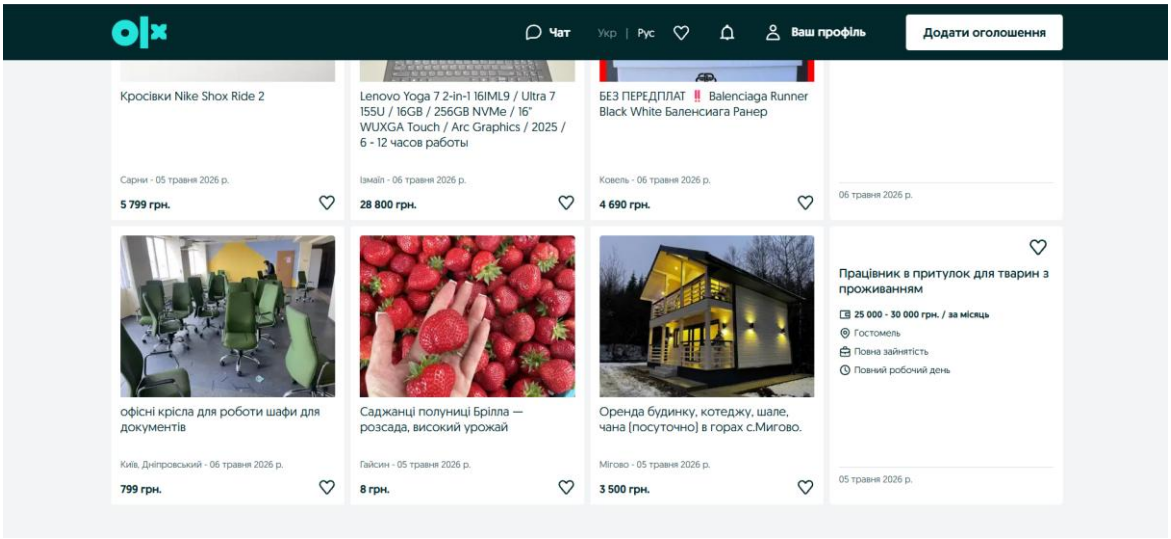


Рисунок 1.3 – Інтерфейс головної сторінки OLX

До переваг OLX можна віднести простий інтерфейс, швидке розміщення оголошень, велику кількість категорій товарів і зручну систему пошуку. Платформа забезпечує швидку взаємодію між користувачами без необхідності створення повноцінного інтернет-магазину.

Разом із цим OLX має і певні недоліки. Інтерфейс містить значну кількість рекламних блоків і платних оголошень, що може ускладнювати пошук необхідних товарів. Також сторінка створення оголошення в OLX має надмірно великі елементи, через що на екрані одночасно відображається невелика кількість інформації (рис. 1.4). Це збільшує довжину сторінки та потребує постійного прокручування під час заповнення оголошення. Крім того, структура форми є вертикально розтягнутою, через що процес додавання товару може займати більше часу. Значна кількість простору між елементами інтерфейсу знижує щільність розміщення інформації та ускладнює швидке сприйняття вмісту сторінки, що може негативно впливати на загальний користувацький досвід взаємодії з платформою.

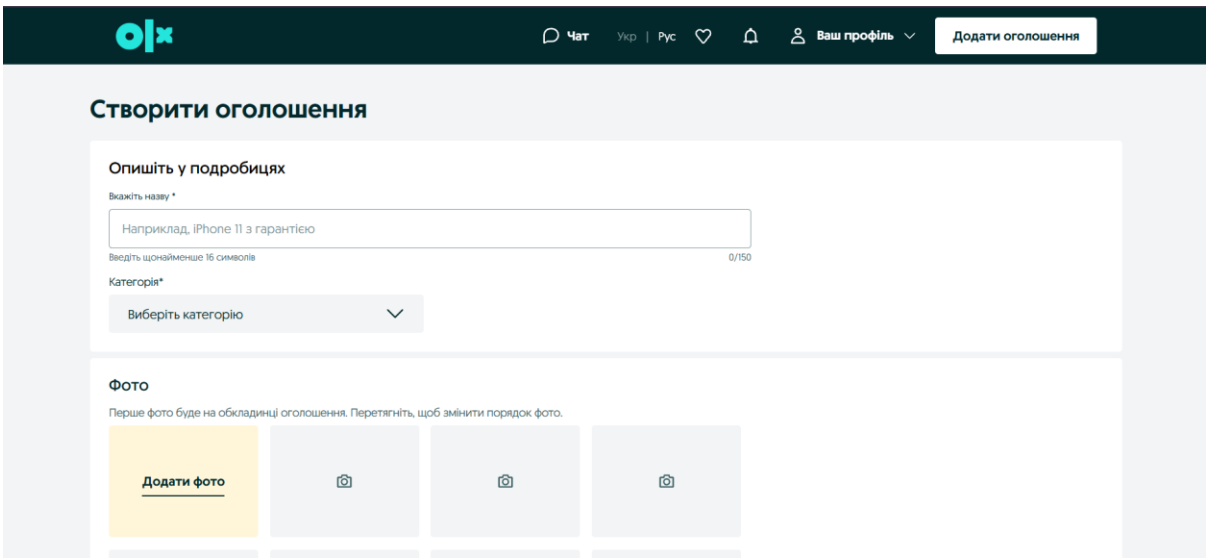


Рисунок 1.4 – Форма додавання оголошення у вебзастосунку OLX

Prom є українською платформою електронної комерції, яка працює за моделлю маркетплейсу та забезпечує взаємодію між продавцями і покупцями. Платформа надає можливість створення власних інтернет-магазинів, розміщення товарів, оформлення замовлень та онлайн-продажу продукції різних категорій. Окрім функцій маркетплейсу, Prom пропонує продавцям інструменти для просування товарів, керування асортиментом та аналітики продажів. Загальний вигляд каталогу товарів платформи Prom наведено на рисунку 1.5.

Prom реалізує широкий набір функціональних можливостей, серед яких пошук і фільтрація товарів, система категорій, особистий кабінет користувача, кошик покупок, система обраних товарів та оформлення замовлень. Крім того, платформа підтримує систему для продавців, що дозволяє продавцям керувати товарами, обробляти замовлення та адмініструвати власний магазин.

До переваг платформи Prom можна віднести адаптивний дизайн, зручну систему пошуку, підтримку великої кількості продавців та простий процес оформлення замовлення. Інтерфейс використовує компактне розташування карток товарів, що дозволяє відображати значну кількість продукції на одному екрані.

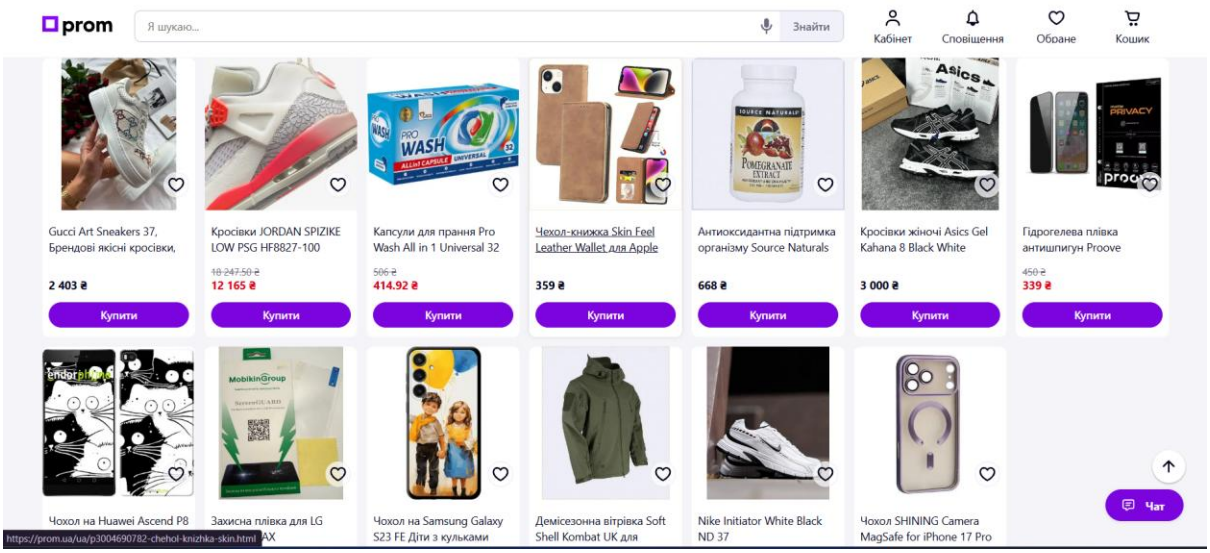


Рисунок 1.5 – Інтерфейс каталогу товарів платформи Prom

Разом із цим платформа має і певні недоліки. Одним із недоліків є розділення системи кошика та обраних товарів на окремі інтерфейсні елементи, що може ускладнювати взаємодію користувача з товарами. Крім того, кошик реалізований у вигляді бічного спливаючого вікна, а не окремої повноцінної сторінки, як показано на рисунку 1.6. Через це користувач отримує обмежений простір для перегляду товарів у кошику, що може ускладнювати редагування великої кількості позицій та перегляд детальної інформації про замовлення.

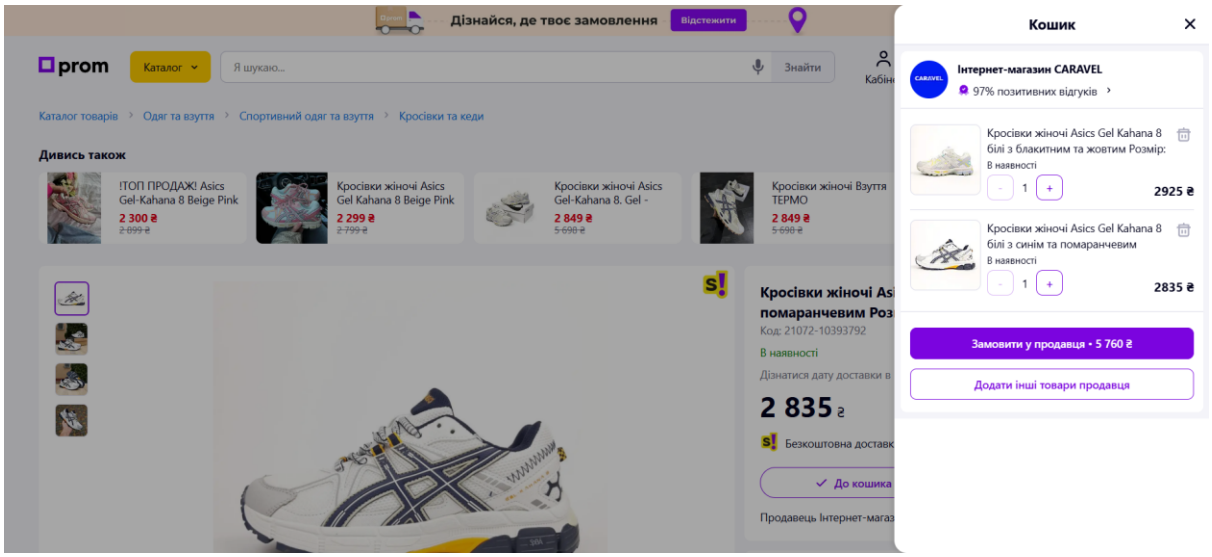


Рисунок 1.6 – Реалізація кошика покупок у вебзастосунку Prom

Також інтерфейс містить значну кількість елементів на сторінці товару, зокрема блоки рекомендацій, рекламні елементи та додаткові пропозиції, що може перевантажувати сторінку та відволікати користувача від основного контенту.

Проведений аналіз показав, що сучасні платформи електронної комерції реалізують широкий набір функцій: пошук і фільтрацію товарів, оформлення замовлень, особисті кабінети, систему для продавців та механізми взаємодії між продавцями і покупцями. Водночас виявлено спільні недоліки: перевантаженість інтерфейсу елементами, складну навігацію, надмірну кількість рекламних блоків та неефективне використання робочого простору сторінок. Окремі платформи також мають недоліки в реалізації кошика покупок, системи створення оголошень та навігаційних компонентів, що ускладнює взаємодію користувача з вебзастосунком і збільшує час виконання основних дій.

1.3 Аналіз сучасних методів побудови вебзастосунків

1.3.1 Клієнт-серверна архітектура вебзастосунків

Однією з найпоширеніших моделей побудови сучасних вебзастосунків є клієнт-серверна архітектура. Її популярність пояснюється активним розвитком мережі Інтернет, збільшенням кількості вебсервісів та необхідністю централізованого зберігання й обробки інформації [9]. Система розподіляється на окремі компоненти, кожен із яких виконує власні функції, що забезпечує ефективнішу організацію роботи вебзастосунку [10].

Клієнтом називають програму або пристрій користувача, який формує запити до сервера для отримання інформації чи виконання операцій [9]. У вебзастосунках цю роль зазвичай виконує браузер: клієнт відображає інтерфейс, надсилає дані користувача та отримує результати обробки запитів [10]. Сервер являє собою програмне або апаратне середовище, що забезпечує

обробку запитів клієнтів, виконання бізнес-логіки та доступ до баз даних [9], приймаючи, опрацьовуючи запити та повертаючи готовий результат клієнтській частині [10]. Один сервер може одночасно обслуговувати велику кількість клієнтів, що дозволяє ефективно використовувати ресурси системи [9].

Взаємодія між клієнтом і сервером зазвичай реалізується за моделлю «запит-відповідь», яка є основою роботи більшості сучасних вебзастосунків [11]: клієнт надсилає HTTP-запит, сервер обробляє дані та повертає відповідь, після чого клієнт оновлює інтерфейс [11]. Наприклад, під час оформлення замовлення клієнтська частина надсилає дані товару та користувача, сервер перевіряє наявність товару й формує відповідь із результатом операції.

Під час проєктування вебзастосунків також широко використовуються архітектурні патерни, зокрема багаторівнева архітектура, що забезпечує гнучкість програмного забезпечення, спрощує супровід системи та покращує повторне використання коду [12]. Питання алгоритмізації процесів формування структурованого контенту також розглядаються в роботах [13].

Клієнт-серверна архітектура має низку переваг: розподіл функціональності дозволяє окремо розвивати клієнтську та серверну частини [10], а централізоване зберігання даних спрощує адміністрування, підвищує безпеку та контроль доступу до інформації [9], що робить таку архітектуру ефективним рішенням для сучасних платформ електронної комерції.

1.3.2 Фронтенд та бекенд вебзастосунків

Сучасні вебзастосунки зазвичай складаються з двох основних частин, якими є фронтенд та бекенд. Вони забезпечують взаємодію користувача із системою та обробку даних на сервері.

Фронтенд є клієнтською частиною вебзастосунку, з якою взаємодіє користувач. Основним завданням фронтенд-розробки є створення зручного та зрозумілого користувацького інтерфейсу, який забезпечує комфортну роботу із системою [14]. До фронтенд-технологій належать HTML, CSS та JavaScript. HTML використовується для створення структури вебсторінки та розміщення її елементів, CSS відповідає за візуальне оформлення інтерфейсу, а JavaScript забезпечує динамічну поведінку сторінок та реалізацію інтерактивних функцій [14]. Зокрема, мова розмітки HTML є основою будь-якої вебсторінки. Вона визначає ієрархічну структуру вмісту та надає браузеру інформацію про те, як організувати та відобразити текст, зображення, посилання та інші елементи [15].

Однією з важливих складових фронтенд-розробки є рендерінг, що є процесом відображення елементів інтерфейсу у браузері користувача. Під час рендерінгу браузер аналізує HTML-документ, застосовує стилі CSS та виконує JavaScript-код для формування фінального вигляду вебсторінки [14]. Завдяки цьому користувач отримує готовий інтерфейс із текстом, зображеннями, кнопками, формами та іншими елементами взаємодії.

Бекенд є серверною частиною вебзастосунку та відповідає за обробку запитів, виконання бізнес-логіки та роботу з даними [16]. На відміну від фронтенд, бекенд-функціональність виконується на сервері та не є безпосередньо доступною користувачу. Серверна частина забезпечує обробку інформації, перевірку даних, керування користувачами та взаємодію з базами даних [16].

Важливою складовою бекенд-частини вебзастосунку є механізми автентифікації та авторизації користувачів, які забезпечують контроль доступу до системи та її функціональних можливостей. Автентифікація використовується для перевірки особи користувача під час входу до системи шляхом підтвердження введених облікових даних [16]. Використання таких механізмів дозволяє обмежити доступ сторонніх осіб до конфіденційної інформації.

1.3.3 REST API

Одним із найпоширеніших підходів до організації взаємодії між клієнтською та серверною частинами вебзастосунку є використання REST API. REST, що є архітектурним стилем побудови вебсервісів, який застосовується для обміну даними між розподіленими системами через мережу Інтернет [17]. У загальному вигляді API виступає посередником між програмними компонентами, визначаючи структуру запитів і відповідей, підтримувані формати даних та механізми обробки помилок [18]. REST API забезпечує взаємодію між різними програмними компонентами за допомогою HTTP-запитів та дозволяє клієнтським застосункам отримувати доступ до ресурсів сервера.

Основною ідеєю REST API є розділення клієнтської та серверної частин системи. У такій архітектурі клієнт відповідає за відображення інтерфейсу та взаємодію з користувачем, тоді як сервер забезпечує обробку запитів, виконання бізнес-логіки та роботу з базами даних [19, 20]. Взаємодія між компонентами здійснюється через URI-адреси ресурсів та HTTP-протокол. Клієнт надсилає запит до сервера, після чого сервер обробляє отримані дані та повертає відповідь у визначеному форматі, наприклад JSON або XML [18], кожен з яких має власні особливості структурування та представлення даних.

Однією з ключових особливостей REST API є stateless-підхід, відповідно до якого кожен HTTP-запит повинен містити всю інформацію, необхідну для його обробки [19, 22]. Сервер не зберігає інформацію про попередні запити користувача між окремими сесіями взаємодії. Такий підхід дозволяє спростити масштабування системи, підвищити продуктивність та забезпечити незалежність між окремими запитами [20].

Для взаємодії між клієнтом і сервером REST API використовує стандартні HTTP-методи. Одним із основних методів є GET, який застосовується для отримання інформації із сервера [21]. Метод POST

використовується для надсилання нових даних та створення ресурсів у системі [21]. Метод PUT дозволяє оновлювати наявні дані або повністю замінювати ресурс новою інформацією [21]. Для видалення ресурсів використовується метод DELETE [21]. Використання стандартних HTTP-методів забезпечує уніфіковану структуру взаємодії між компонентами системи та спрощує інтеграцію вебзастосунків із зовнішніми сервісами.

Отже, сучасні методи побудови вебзастосунків базуються на використанні клієнт-серверної архітектури, розподілі функціональності між фронтенд та бекенд-компонентами, а також застосуванні REST API для взаємодії між частинами системи. Сучасні технології сприяють створенню інтерактивних користувацьких інтерфейсів та підтримці роботи системи на різних типах пристроїв.

1.4 Постановка задачі

Розроблення клієнт-серверних вебзастосунків електронної комерції із зрозумілим інтерфейсом та ефективною взаємодією між продавцями і покупцями є актуальним завданням в умовах стрімкого зростання онлайн-торгівлі. Зростаюча конкуренція між онлайн-платформами спонукає до створення рішень, що поєднують технічну надійність із зручністю використання. Тому ставиться завдання створити вебзастосунок, що забезпечить повноцінний цикл електронної комерції із використанням сучасних вебтехнологій.

Об’єктом роботи є процес організації електронної комерції у форматі взаємодії між продавцями та покупцями через вебзастосунок.

Метою роботи є розроблення клієнт-серверного вебзастосунку електронної комерції, який забезпечує взаємодію між продавцями та покупцями, підтримує управління товарами, оформлення замовлень та роботу користувацького інтерфейсу.

Для досягнення мети необхідно вирішити такі завдання:

– спроєктувати структуру клієнтської та серверної частин вебзастосунку;

– спроєктувати структуру бази даних та взаємозв’язки між основними сутностями системи;

– реалізувати систему реєстрації, авторизації та керування ролями користувачів;

– реалізувати функціонал перегляду товарів, пошуку, фільтрації та роботи з каталогом;

– реалізувати систему кошика покупок та оформлення замовлень;

– реалізувати систему для продавців для додавання та керування товарами;

– забезпечити зберігання та обробку зображень товарів із використанням зовнішніх сервісів;

– реалізувати систему електронних повідомлень після реєстрації та оформлення замовлення;

– забезпечити взаємодію між клієнтською та серверною частинами вебзастосунку через API;

– провести тестування основних функціональних можливостей системи та проаналізувати отримані результати.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Обґрунтування архітектури вебзастосунку

Архітектура вебзастосунку визначає принципи взаємодії між його компонентами, порядок обробки даних та можливості подальшого розширення функціоналу. Ефективна організація таких процесів потребує застосування відповідних технологій підтримки прийняття рішень в інформаційних системах [22]. При проєктуванні складних програмних систем доцільно спиратися на принцип декомпозиції, відповідно до якого система поділяється на окремі підсистеми з чітко визначеними вхідними та вихідними зв’язками між ними [23]. Реалізація цього принципу дозволяє модифікувати окремі компоненти без впливу на суміжні частини системи та забезпечує гнучкість подальшого розвитку. Питання моделювання та оцінювання станів складних об’єктів і систем розглядаються також у роботах [24].

Клієнтська частина відповідає за відображення інтерфейсу користувача та забезпечення взаємодії з основними функціями платформи. Через неї користувач може переглядати каталог товарів, відкривати сторінку окремого товару, працювати з кошиком і списком бажаного, проходити реєстрацію та авторизацію, оформлювати замовлення, а також подавати заявку на отримання статусу продавця. Для користувача зі статусом продавця клієнтська частина також забезпечує доступ до сторінок управління товарами. Таким чином, клієнтська частина охоплює весь спектр взаємодій кінцевого користувача із системою та формує його цілісний досвід роботи з платформою.

Серверна частина виконує роль проміжного рівня між клієнтською частиною, базою даних та зовнішніми сервісами. Вона приймає запити від клієнта, перевіряє коректність отриманих даних, виконує операції відповідно до бізнес-логіки та повертає результат обробки у структурованому вигляді.

Розміщення основної логіки на серверній стороні підвищує контрольованість роботи системи та спрощує подальше супроводження вебзастосунку. При розробці серверної та клієнтської частин дотримуються відповідних стандартів оформлення коду для обраних мов програмування [25]. Зосередження бізнес-логіки на сервері також унеможливлює несанкціонований доступ до критичних операцій шляхом маніпуляцій із клієнтською частиною.

Взаємодія між клієнтською та серверною частинами здійснюється через програмний інтерфейс, побудований за принципами REST API. Такий підхід дозволяє представити основні функції системи окремими маршрутами, зокрема для реєстрації, авторизації, роботи з товарами, оформлення замовлення та подання заявки на отримання статусу продавця. Це робить структуру серверної частини зрозумілою та придатною для подальшого розвитку.

Кожен маршрут серверної частини відповідає певній сутності та операції, що дозволяє дотримуватися принципів REST щодо узгодженого використання HTTP-методів. Для роботи з товарами передбачено маршрути отримання переліку товарів, отримання інформації про окремий товар за ідентифікатором, додавання нового товару, оновлення існуючого товару та його видалення. Метод GET застосовується для отримання даних, POST для створення нових записів, PUT для оновлення наявних, а DELETE для видалення товарів із колекції товарів.

Для роботи з обліковими записами користувачів передбачено окрему групу маршрутів, які забезпечують реєстрацію нового користувача, авторизацію за електронною адресою та паролем, а також отримання даних поточного користувача. Реєстрація передбачає перевірку унікальності електронної адреси та збереження зашифрованого пароля в колекції користувачів. Під час авторизації введений пароль порівнюється зі збереженим зашифрованим значенням, після чого формується відповідь із

даними користувача, необхідними для подальшої роботи клієнтської частини.

Окрему групу становлять маршрути, пов’язані з оформленням замовлень. Під час створення замовлення серверна частина отримує перелік товарів, відомості про доставку та спосіб оплати, формує новий документ у колекції замовлень та повертає клієнту підтвердження. Додатково передбачено маршрут отримання історії замовлень конкретного користувача, що дозволяє відображати на сторінці профілю перелік попередніх покупок.

Маршрути, що стосуються подання заявки на отримання статусу продавця, дозволяють користувачу надіслати дані про магазин на сервер, де вони зберігаються у колекції продавців, а полю продавець у документі користувача присвоюється значення «істина». Після цього клієнтська частина отримує доступ до сторінок управління товарами, доступних лише користувачам зі статусом продавця.

Групування маршрутів за функціональним призначенням відповідає прийнятим практикам побудови REST API, згідно з якими кожен ресурс системи описується власним набором операцій [25]. Такий підхід полегшує тестування окремих частин серверної логіки, оскільки кожен маршрут може перевірятися незалежно від інших, а також спрощує додавання нових функцій без необхідності змінювати вже реалізовані маршрути. Крім того, чітке групування полегшує навігацію по коду під час командної розробки та повторного використання маршрутів у суміжних проєктах.

Для збереження основних даних вебзастосунку доцільно використати Firebase Firestore. Цей сервіс має документоорієнтовану структуру, що відповідає спроєктованій моделі бази даних із колекціями та документами [26]. Використання такого підходу спрощує організацію даних, їх оновлення та отримання на серверній частині. Подібний підхід до організації серверної частини на основі Firebase Firestore застосовується також під час розроблення інших вебзастосунків, орієнтованих на роботу з документоорієнтованими даними [26].

Окремим компонентом архітектури є сервіс надсилання електронних повідомлень. Він використовується для інформування користувачів про важливі дії в системі, зокрема про успішну реєстрацію або оформлення замовлення. Автоматичне надсилання повідомлень підвищує рівень залученості користувачів і забезпечує прозорість виконання операцій.

У межах архітектури також передбачено розмежування доступу за ролями користувачів. Це дозволяє відокремити функції покупця від функцій продавця та забезпечити доступ до відповідних сторінок і операцій залежно від статусу користувача. Перевірка ролі виконується як на клієнтській стороні під час відображення елементів інтерфейсу, так і на серверній стороні під час обробки запитів, що унеможливлює несанкціонований доступ до захищених маршрутів.

Таким чином, клієнт-серверна архітектура є доцільною для розроблення вебзастосунку, оскільки вона забезпечує логічне розділення функцій між компонентами системи, спрощує обробку запитів, дозволяє використовувати базу даних і зовнішні сервіси, а також створює основу для подальшого розширення функціоналу.

2.2 Моделювання функціональної структури системи

Функціональна структура вебзастосунку визначає основні можливості системи, ролі користувачів та порядок їх взаємодії з функціональними модулями. На етапі проєктування доцільно виділити основні типи користувачів, які взаємодіють із системою, а також визначити набір функцій, доступних кожному з них.

У межах вебзастосунку передбачено три основні типи взаємодії із системою: гість, покупець та продавець. Гість є неавторизованим користувачем, який має обмежений доступ до функціоналу платформи. Він може переглядати каталог товарів, додавати товари до кошика, додавати

товари до списку бажаного, редагувати вміст кошика та перейти до сторінки реєстрації. Такий рівень доступу дозволяє користувачу ознайомитися з товарами без обов'язкового створення облікового запису, але не надає повного доступу до процесу оформлення замовлення.

Покупець є авторизованим користувачем системи, який має доступ до основного функціоналу електронної комерції. Він може увійти до системи, оформлювати замовлення та подати заявку на отримання статусу продавця, якщо він планує розміщувати власні товари на платформі.

Продавець є користувачем, який отримав розширені права доступу після подання відповідної заявки. Для цієї ролі передбачено функції перегляду власних товарів, додавання нових товарів, редагування інформації про товари, видалення товарів та публікації товарів у каталозі. Наявність окремої ролі продавця дозволяє розмежувати можливості звичайного покупця та користувача, який наповнює платформу товарами.

Функціональну структуру вебзастосунку доцільно подати за допомогою діаграми варіантів використання UML. Така діаграма дозволяє відобразити, які дії можуть виконувати різні користувачі системи [27]. Діаграму варіантів використання вебзастосунку наведено на рисунку 2.1.

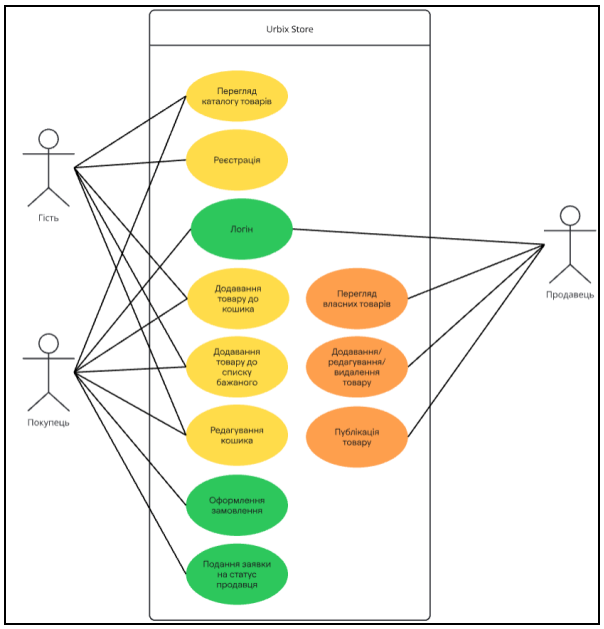


Рисунок 2.1 – Діаграма варіантів використання вебзастосунку

Запропонована функціональна структура дозволяє чітко розділити можливості різних категорій користувачів. Для неавторизованого користувача система виконує ознайомчу функцію, для покупця забезпечує процес вибору та придбання товарів, а для продавця надає інструменти управління товарними позиціями. Такий поділ функцій є доцільним для вебзастосунку електронної комерції, оскільки він забезпечує логіку доступу, спрощує подальшу реалізацію модулів і створює основу для розширення системи в майбутньому.

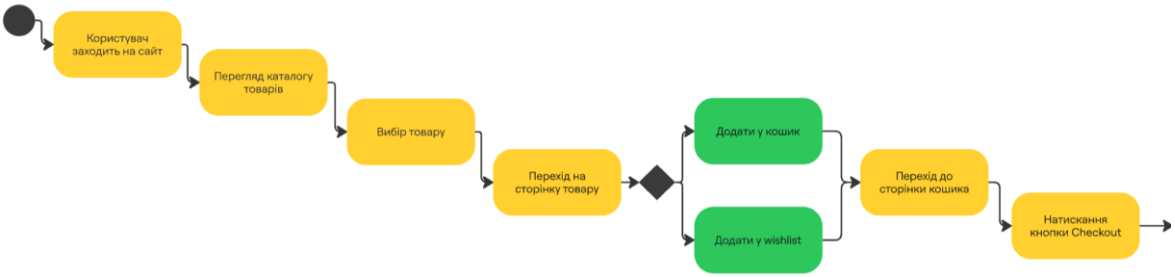
2.3 Проєктування бізнес-процесів вебзастосунку

Проєктування бізнес-процесів вебзастосунку дозволяє визначити послідовність дій користувачів, умови переходу між етапами та взаємодію основних функціональних модулів. Для відображення таких процесів доцільно використати діаграми діяльності UML, оскільки вони дають змогу наочно подати логіку виконання дій та можливі розгалуження [27].

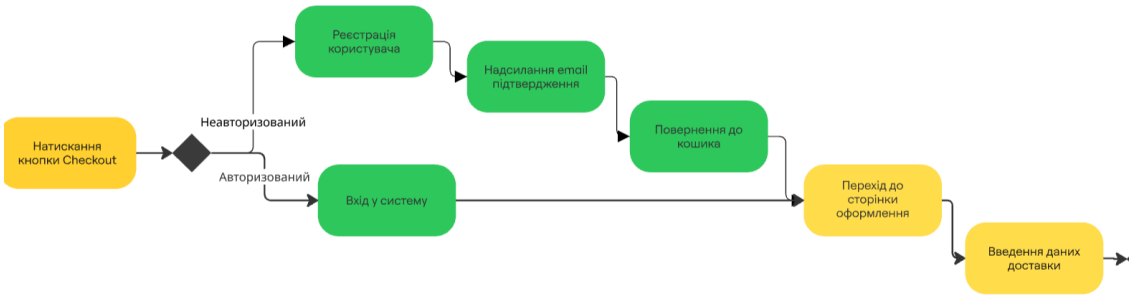
Для вебзастосунку розглянуто два основні бізнес-процеси: оформлення замовлення покупцем та реєстрацію користувача як продавця з подальшим управлінням товарами. Діаграма діяльності процесу оформлення замовлення складається з трьох частин. Перша частина охоплює початковий етап взаємодії користувача з платформою. Після формування кошика користувач має змогу переглянути його вміст, змінити кількість товарів або видалити окремі позиції. Діаграму діяльності процесу вибору товару та формування кошика показано на рисунку 2.2 а).

Друга частина відображає етап ідентифікації користувача перед переходом до оформлення замовлення. Якщо користувач не авторизований, система перенаправляє його на сторінку входу або реєстрації. Діаграму діяльності процесу авторизації користувача та переходу до оформлення показано на рисунку 2.2 б).

Третя частина описує безпосередньо процес оформлення замовлення, який включає введення даних для доставки, вибір способу оплати та підтвердження замовлення. Діаграму діяльності процесу оформлення замовлення показано на рисунку 2.2 в).



а)



б)

в)

Рисунок 2.2 – Діаграма діяльності процесу оформлення замовлення:
а) вибір товару та формування кошика; б) авторизація користувача та перехід до оформлення; в) оформлення замовлення

На початковому етапі користувач переглядає каталог, відкриває сторінку товару та додає потрібні позиції до кошика або списку бажаного. Після завершення вибору він переходить до кошика, де може переглянути додані товари, змінити їх кількість або видалити окремі позиції.

Перед переходом до оформлення замовлення система перевіряє, чи користувач авторизований. Якщо авторизація відсутня, користувач перенаправляється на сторінку входу або реєстрації. Після успішної авторизації він продовжує оформлення замовлення.

На сторінці оформлення користувач вводить дані доставки, обирає спосіб оплати та підтверджує замовлення. У разі помилки при заповненні форми користувач повертається до неї для виправлення введених даних, що унеможливлює збереження некоректного замовлення.

Після успішної перевірки формується замовлення, яке передається на серверну частину для збереження в базі даних. Далі користувачу надсилається електронне повідомлення з підтвердженням замовлення, кошик очищується, а на екрані відображається сторінка успішного оформлення.

Другим важливим бізнес-процесом є реєстрація користувача як продавця та подальше управління товарами. Цей процес дозволяє розширити можливості користувача та надати йому доступ до функцій додавання, редагування, видалення і публікації товарів. Відокремлення процесу отримання статусу продавця від звичайної реєстрації забезпечує контроль над складом учасників платформи.

Перша частина діаграми відображає процес подання заявки на отримання статусу продавця. Користувач заповнює відповідну форму із зазначенням необхідних даних та надсилає заявку на розгляд. Після перевірки система надає користувачу статус продавця та розширює його права доступу. Діаграму діяльності процесу реєстрації користувача як продавця показано на рисунку 2.3 а).

Друга частина охоплює основні дії продавця з управління товарами після отримання відповідного статусу. Продавець може додавати нові товари

із зазначенням назви, опису, ціни та зображень, редагувати існуючі позиції, змінювати їх статус публікації або видаляти товари з платформи. Кожна з цих дій обробляється системою та відображається в каталозі відповідно до встановлених правил. Діаграму діяльності процесу управління товарами продавця показано на рисунку 2.3 б).

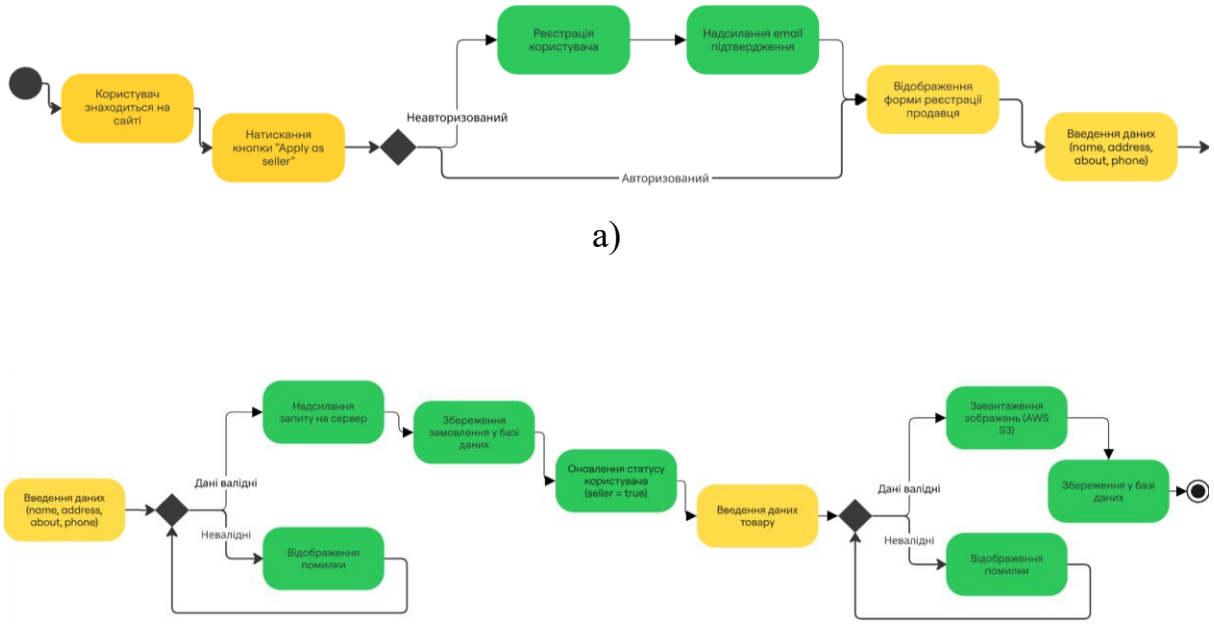


Рисунок 2.3 – Діаграма діяльності процесу реєстрації та роботи продавця в системі: а) реєстрація користувача як продавця; б) управління товарами продавця

Після авторизації користувач заповнює форму реєстрації продавця, у якій зазначає дані про магазин або свою діяльність. Система перевіряє коректність заповнення форми, зокрема наявність обов’язкових полів, допустиму довжину текстових значень, формат номера телефону та підтвердження згоди з умовами використання платформи. У разі виявлення некоректних даних система повертає користувача до форми з відповідним повідомленням про помилку. Після успішного проходження всіх перевірок дані про магазин зберігаються у базі даних, а обліковий запис користувача

оновлюється відповідно до нового статусу. У разі успішної перевірки користувач отримує доступ до функцій продавця, що відображається зміною навігаційної панелі та появою додаткових сторінок управління товарами.

У межах управління товарами продавець може переглядати власні товари, додавати нові позиції, редагувати або видаляти вже створені товари. Перелік власних товарів відображається у вигляді карток із короткою інформацією про кожну позицію, що дозволяє продавцю швидко орієнтуватися в асортименті та переходити до редагування потрібного товару. Під час додавання товару вводяться його назва, описи, ціна, розмір знижки, кількість на складі, категорія та зображення. Система автоматично розраховує кінцеву ціну з урахуванням зазначеної знижки, що зменшує ймовірність помилок при встановленні вартості товару.

Зображення товару завантажуються безпосередньо до хмарного сховища, а у базі даних зберігаються лише посилання на них, що забезпечує оптимальну продуктивність системи. Також передбачено можливість збереження товару як чернетки або його публікації в каталозі. Режим чернетки дозволяє продавцю підготувати товар до публікації заздалегідь, не роблячи його видимим для покупців до готовності. Після публікації товар одразу з’являється у відповідній категорії каталогу та стає доступним для пошуку покупцями.

Таким чином, проєктування бізнес-процесів вебзастосунку дозволяє визначити основні сценарії роботи системи: оформлення замовлення покупцем та отримання користувачем доступу до функцій продавця.

2.4 Проєктування структури бази даних

Проєктування структури бази даних є важливим етапом розроблення вебзастосунку, оскільки база даних забезпечує збереження основної інформації, необхідної для роботи системи. Від коректності структури даних

залежить продуктивність виконання запитів, цілісність інформації та зручність подальшого масштабування системи.

Для вебзастосунку доцільно використати документоорієнтовану структуру бази даних, у якій інформація зберігається у вигляді колекцій та документів. Назви колекцій та полів обирались відповідно до принципів чистого коду, що передбачають використання зрозумілих імен, які точно відображають суть даних [28]. Такий підхід відповідає особливостям системи електронної комерції, де основні сутності мають різну структуру, але логічно пов’язані між собою. У межах проєктування бази даних виділено такі основні колекції: користувачів (users), продавців (sellers), товарів (products) та замовлень (orders).

Загальна структура бази даних формується відповідно до функціональних модулів вебзастосунку. Дані користувачів зберігаються окремо від даних продавців, товари пов’язуються з відповідним продавцем, а оформлені замовлення пов’язуються з покупцем. Такий поділ дозволяє уникнути змішування різних типів інформації та забезпечити зручну обробку даних на серверній частині. Загальну структуру колекцій та полів бази даних вебзастосунку наведено на рисунку 2.4.

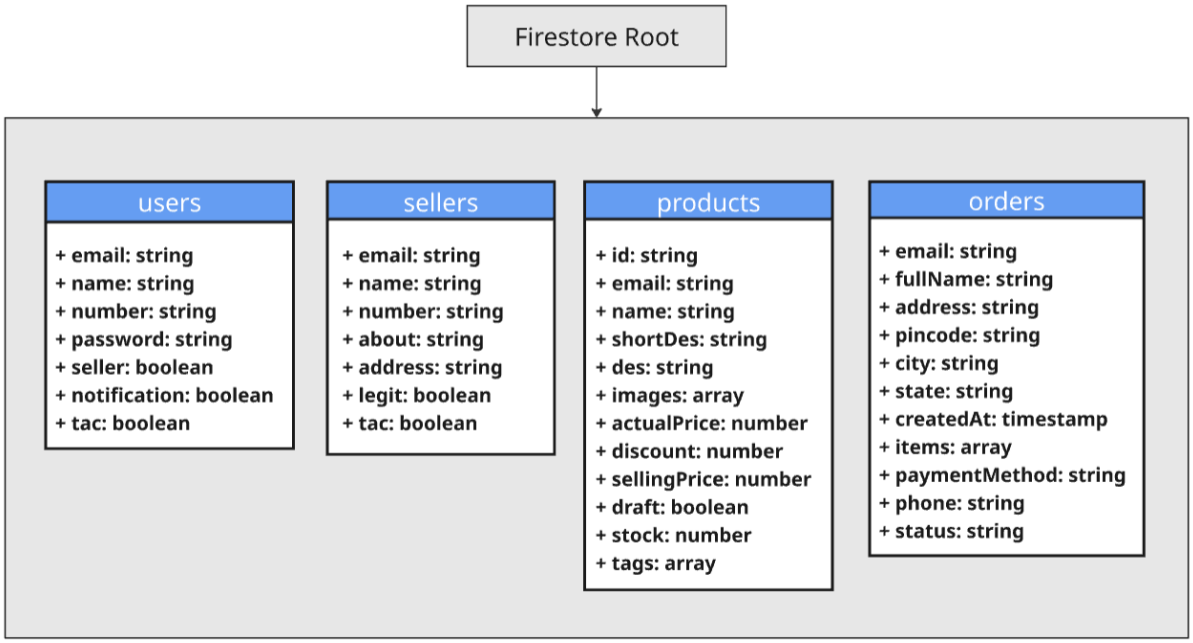


Рисунок 2.4 – Структура колекцій та полів бази даних вебзастосунку

Колекція користувачів призначена для зберігання даних зареєстрованих користувачів. До її основних полів належать електронна адреса, ім’я користувача, номер телефону, зашифрований пароль та ознака наявності статусу продавця. Поле продавця має логічний тип і визначає, чи має користувач доступ до функціоналу продавця. Використання електронної адреси як ідентифікатора документа забезпечує швидкий пошук під час авторизації та унеможливлює реєстрацію двох акаунтів з однаковою поштою.

Колекція продавців використовується для зберігання додаткової інформації про користувачів, які отримали статус продавця. У ній зберігаються назва магазину, електронна адреса продавця, контактний номер телефону, адреса та опис діяльності. Ці дані використовуються для ідентифікації продавця та відображення інформації про його магазин. Відокремлення профілю продавця від профілю користувача дозволяє гнучко керувати правами доступу та зберігати лише ті поля, які є релевантними для кожної з ролей.

Колекція товарів призначена для зберігання інформації про товари, які додаються продавцями. Кожен документ у цій колекції описує окремий товар і містить назву, короткий та повний опис, початкову ціну, розмір знижки, кінцеву ціну, кількість товару на складі, категорію, теги, електронну адресу продавця та масив посилань на зображення. Зв’язок товару з продавцем забезпечується через електронну адресу, що дозволяє відображати на сторінці управління лише товари конкретного продавця.

У структурі товару зберігаються не самі зображення, а масив посилань на них. Це дозволяє не перевантажувати базу даних медіафайлами та використовувати ці посилання для відображення товарів у каталозі й на сторінці детального перегляду. Такий підхід відповідає принципу розділення відповідальності між сервісами: база даних відповідає за структуровані дані, тоді як медіафайли обробляються окремим хмарним сховищем.

Колекція замовлень призначена для зберігання оформлених замовлень. Кожен документ містить інформацію про покупця, перелік товарів, загальну

суму замовлення, відомості про доставку, спосіб оплати, дату створення та статус замовлення. Перелік товарів зберігається у вигляді масиву, оскільки одне замовлення може містити кілька товарних позицій. Наявність поля статусу дозволяє у майбутньому розширити систему обробки замовлень, додавши етапи підтвердження, відправки та доставки.

Зв’язки між основними сутностями бази даних будуються на основі електронної адреси користувача або продавця. Користувач може мати один профіль у колекції користувачів. Після отримання статусу продавця для нього створюється додатковий запис у колекції продавців. Продавець може мати багато товарів у колекції товарів, а покупець може оформлювати багато замовлень у колекції замовлень.

Таким чином, запропонована структура бази даних відображає основні сутності вебзастосунку та зв’язки між ними. Вона є достатньою для реалізації реєстрації користувачів, роботи продавців, управління товарами та оформлення замовлень.

2.5 Обґрунтування вибору технологій та сервісів

2.5.1 Обґрунтування вибору технологій клієнтської частини

Клієнтська частина вебзастосунку відповідає за відображення інтерфейсу користувача та забезпечення взаємодії з основними функціями системи. Сучасна фронтенд-розробка пропонує широкий спектр інструментів, від JavaScript-фреймворків, таких як Angular, React та Vue.js, до нативних веб-технологій [29, 30]. Фреймворки є доцільними для великих корпоративних застосунків зі складною компонентною структурою, проте несуть додаткові залежності, потребують налаштування систем збирання та підвищують поріг входження.

Для вебзастосунку Urbix Store обрано нативний підхід на основі HTML, CSS та JavaScript, оскільки він забезпечує менший обсяг залежностей,

спрощене розгортання та повний контроль над кодом без накладних витрат. HTML забезпечує структуру сторінок, CSS відповідає за оформлення інтерфейсу, а JavaScript використовується для реалізації динамічної поведінки, зокрема перевірки форм, оновлення кошика, розрахунку вартості замовлення та взаємодії з серверною частиною. Використання нативних технологій також знижує ризик порушення роботи застосунку через несумісність версій сторонніх бібліотек.

Для обміну даними між клієнтською та серверною частинами доцільно використати браузерний інтерфейс запитів «fetch». Він дозволяє надсилати запити до серверних маршрутів та отримувати відповіді у форматі JSON без повного перезавантаження сторінки, що є зручним для реалізації основних сценаріїв роботи вебзастосунку. Асинхронна природа цього інтерфейсу дозволяє виконувати мережеві операції паралельно з рендерингом інтерфейсу, що позитивно впливає на швидкість відгуку системи.

Для тимчасового збереження даних на стороні користувача доцільно використати LocalStorage. Це браузерне сховище застосовується для збереження інформації про кошик, список бажаного та поточного користувача, що забезпечує збереження частини даних між переходами між сторінками. Збереження даних кошика на стороні клієнта дозволяє гостю формувати замовлення ще до реєстрації, що знижує бар’єр для здійснення покупки.

Файлова структура клієнтської частини організована за принципом розділення сторінок, стилів та сценаріїв. Кожна сторінка вебзастосунку має власний HTML-файл, що відображає її розмітку, тоді як стилі винесено до окремих CSS-файлів, а логіка взаємодії до відповідних файлів JavaScript. Такий поділ відповідає принципу розділення відповідальності та полегшує внесення змін до окремих частин інтерфейсу без впливу на інші сторінки.

Для оформлення інтерфейсу застосовано загальні стилі, що визначають кольорову палітру, типографіку та розташування основних елементів, а також стилі, специфічні для окремих сторінок. Такий підхід дозволяє

підтримувати єдиний візуальний стиль вебзастосунку та водночас адаптувати оформлення окремих сторінок відповідно до їх призначення. Спільні елементи інтерфейсу, зокрема навігаційне меню та нижній блок, реалізовано як повторювані блоки розмітки, що використовуються на всіх сторінках вебзастосунку.

JavaScript-файли клієнтської частини організовано відповідно до функціонального призначення сторінок: окремий файл відповідає за логіку каталогу товарів, інший за роботу з кошиком, ще один за оформлення замовлення. Така організація коду відповідає принципам модульності та полегшує супроводження вебзастосунку, оскільки зміни в логіці однієї сторінки не впливають на код інших сторінок.

2.5.2 Обґрунтування вибору технологій серверної частини

Серверна частина вебзастосунку призначена для обробки запитів клієнтської частини, виконання основної логіки системи, перевірки даних, взаємодії з базою даних та зовнішніми сервісами.

Для розроблення серверної частини доцільно використати Node.js. Ця платформа дозволяє виконувати JavaScript на стороні сервера, що забезпечує використання однієї мови програмування для клієнтської та серверної частин. Такий підхід спрощує розроблення, оскільки логіка роботи системи може бути реалізована в єдиному мовному середовищі. Крім того, Node.js підтримує асинхронну обробку запитів, що є важливим для вебзастосунку, у якому одночасно можуть виконуватися різні операції користувачів [31]. Висока продуктивність платформи при роботі з великою кількістю одночасних підключень робить її особливо придатною для застосунків електронної комерції.

Для побудови серверної логіки доцільно використати Express. Він дозволяє організувати маршрутизацію запитів, обробку даних, передавання

відповідей клієнтській частині та створення окремих маршрутів для кожної функції системи. Мінімалістичний підхід цього фреймворку не нав’язує жорсткої структури проєкту, надаючи розробнику свободу у виборі архітектурних рішень відповідно до потреб конкретного застосунку.

Для взаємодії між клієнтською та серверною частинами доцільно використати REST API. Такий підхід дозволяє структурувати обмін даними через окремі маршрути та стандартні HTTP-запити. Для вебзастосунку електронної комерції це є зручним рішенням, оскільки кожна дія користувача може відповідати окремому запиту до серверної частини [32].

Для захисту паролів користувачів доцільно використати бібліотеку шифрування «bcrypt». Оскільки пароль є конфіденційною інформацією, його не можна зберігати у відкритому вигляді. Використання цієї бібліотеки дозволяє перетворювати пароль у зашифроване значення перед збереженням у базі даних. Під час входу до системи введений пароль порівнюється з уже збереженим зашифрованим значенням. Такий підхід підвищує безпеку облікових записів користувачів і є стандартною практикою у сучасних вебзастосунках.

Для роботи з конфіденційними параметрами доцільно використати бібліотеку «dotenv». Цей інструмент дозволяє зберігати службові дані, ключі доступу та інші змінні середовища окремо від основного програмного коду. Це є важливим для безпеки, оскільки конфіденційні значення не повинні бути відкрито розміщені в коді застосунку [33]. Використання файлу середовища також спрощує перемикання між конфігураціями розробки та виробничого середовища без зміни програмного коду.

Файлова структура серверної частини організована відповідно до функціональних модулів вебзастосунку. Маршрути, що обробляють запити, винесено до окремих файлів за принципом приналежності до певної сутності: товари, користувачі, замовлення та заявки на отримання статусу продавця. Кожен файл маршрутів підключається до основного файлу застосунку через

систему модулів Node.js, що дозволяє підтримувати структуру серверної частини у впорядкованому вигляді та полегшує її подальше розширення.

Окремо реалізовано модуль конфігурації, який відповідає за встановлення з'єднання з Firebase Firestore та ініціалізацію клієнта для роботи з Amazon S3. Цей модуль завантажується під час запуску серверної частини та надає іншим частинам застосунку доступ до бази даних і хмарного сховища без повторної ініціалізації. Винесення конфігураційної логіки в окремий модуль унеможливлює дублювання коду підключення та гарантує, що всі частини застосунку працюють з єдиним екземпляром клієнта бази даних. Логіка надсилання електронних повідомлень через Nodemailer також виокремлена в самостійний модуль, який викликається з маршрутів реєстрації та оформлення замовлення. Такий підхід дозволяє легко змінювати шаблони повідомлень або замінювати поштовий сервіс без втручання в логіку основних маршрутів.

Таке розділення серверної частини на модулі маршрутів, конфігурації та допоміжних сервісів відповідає принципам модульної архітектури програмного забезпечення [25]. Кожен модуль виконує чітко визначену функцію та має мінімальну кількість залежностей від інших частин системи, що відповідає принципу єдиної відповідальності. Це підвищує читабельність коду, спрощує тестування окремих модулів та полегшує внесення змін під час подальшого розвитку вебзастосунку.

2.5.3 Обґрунтування вибору бази даних та зовнішніх сервісів

Для збереження основних даних вебзастосунку доцільно використати Firebase Firestore. Цей сервіс має документоорієнтовану структуру, що відповідає спроєктованій моделі бази даних із колекціями та документами. Використання такого підходу спрощує організацію даних, їх оновлення та отримання на серверній частині. Гнучкість схеми документів дозволяє

вносити зміни до структури даних без необхідності виконання складних міграцій, що є суттєвою перевагою на початкових етапах розроблення.

Firebase Firestore також є доцільним рішенням через те, що не потребує самостійного розгортання окремого сервера бази даних. Це спрощує процес розроблення та дозволяє зосередитися на реалізації функціоналу вебзастосунку. Крім того, сервіс підтримує подальше збільшення обсягу даних, що є важливим для системи електронної комерції. Вбудовані механізми автентифікації та правила безпеки Firebase дозволяють тонко налаштовувати права доступу до окремих колекцій і документів.

Для зберігання зображень товарів доцільно використати Amazon S3, оскільки цей сервіс призначений для розміщення медіафайлів у хмарному сховищі та може використовуватися разом із базою даних, у якій зберігаються лише посилання на файли. Висока надійність та доступність цього сервісу забезпечують стабільне відображення зображень товарів незалежно від навантаження на основний сервер застосунку.

Для надсилання електронних повідомлень користувачам доцільно використати Nodemailer. Цей інструмент дозволяє реалізувати автоматичне інформування користувачів після виконання важливих дій у системі, зокрема після реєстрації або оформлення замовлення. Підтримка різних поштових протоколів і гнучке налаштування шаблонів повідомлень роблять цей інструмент універсальним рішенням для реалізації поштових сповіщень у вебзастосунках.

Таким чином, Firebase Firestore, Amazon S3 та Nodemailer відповідають функціональним потребам вебзастосунку. Firebase Firestore забезпечує збереження основних даних, Amazon S3 використовується для роботи із зображеннями товарів, а Nodemailer забезпечує електронне інформування користувачів. Спільне використання цих сервісів дозволяє побудувати надійну та масштабовану інфраструктуру вебзастосунку без необхідності самостійно розгортати і підтримувати серверні компоненти для кожної з цих функцій.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Реалізація серверної частини вебзастосунку

3.1.1 Налаштування серверного середовища

Налаштування серверного середовища виконано у головному серверному файлі застосунку. На початку файлу підключено необхідні модулі для роботи серверної частини, зокрема Express, Firebase Admin, «bcrypt», «path», «fs», Nodemailer, AWS SDK та «dotenv».

Express використовується для створення серверного застосунку, оголошення маршрутів та обробки HTTP-запитів. Модуль «path» застосовується для формування шляхів до файлів клієнтської частини, які розміщені в папці «public». Модуль «fs» використовується для зчитування службового файлу Firebase, а Firebase Admin забезпечує доступ серверної частини до Firestore.

Підключення до бази даних виконується через службовий файл Firebase. Після зчитування цього файлу здійснюється ініціалізація Firebase Admin і створюється змінна «db», через яку надалі виконуються операції з колекціями Firestore. Для роботи з конфіденційними параметрами використано «dotenv». Завдяки цьому ключі доступу до Amazon S3 та дані для надсилання електронної пошти зберігаються у змінних середовища, а не безпосередньо в основному коді програми.

У серверному файлі також налаштовано статичну папку «public», з якої користувачу надсилаються HTML-сторінки вебзастосунку. Для обробки даних у форматі JSON використано проміжне програмне забезпечення «express.json()», що дозволяє серверу отримувати дані з тіла запиту.

Лістинг 3.1 Налаштування серверного середовища:

```
const express = require('express');
const fs = require('fs');
```

```
...
dotenv.config();
const serviceAccountJSON = fs.readFileSync(serviceAccountPath, 'utf8');
const serviceAccount = JSON.parse(serviceAccountJSON);
admin.initializeApp({
   credential: admin.credential.cert(serviceAccount),
   ignoreUndefinedProperties: true
});
let db = admin.firestore();
let staticPath = path.join(__dirname, 'public');
const app = express();
app.use(express.static(staticPath));
app.use(express.json());
```

Після налаштування основних модулів сервер запускається на порту 3000. Для переходу користувача за неіснуючим маршрутом реалізовано перенаправлення на сторінку 404, що дозволяє коректно обробляти помилкові запити.

3.1.2 Реалізація реєстрації та авторизації користувачів

Реєстрація користувачів реалізована за допомогою маршрутів «/signup». GET-запит до цього маршруту повертає сторінку реєстрації, а POST-запит обробляє дані, введені користувачем у форму. Під час реєстрації сервер отримує ім'я користувача, електронну адресу, пароль, номер телефону, підтвердження умов використання та згоду на отримання повідомлень.

Перед збереженням користувача у базі даних виконується перевірка введених даних. Сервер перевіряє довжину імені, наявність електронної

адреси, довжину пароля, наявність номера телефону, коректність номера та підтвердження умов використання. Якщо хоча б одна умова не виконується, сервер повертає клієнтській частині повідомлення про помилку у форматі JSON.

Після успішної перевірки виконується пошук користувача в колекції користувачів за електронною адресою. Якщо документ із такою електронною адресою вже існує, система повертає повідомлення про те, що користувач уже зареєстрований. Якщо такого документа немає, пароль користувача зашифровується за допомогою бібліотеки «bcrypt» шляхом генерації унікального значення. Після цього дані користувача зберігаються у Firestore, а клієнтська частина отримує відповідь із основною інформацією про новостворений обліковий запис.

Лістинг 3.2 Реалізація хешування пароля під час реєстрації користувача:

```
db.collection('users').doc(email).get()
.then(user => {
if (user.exists) {
return res.json({ 'alert': 'Email Already Exists' });
} else {
bcrypt.genSalt(10, (err, salt) => {
bcrypt.hash(password, salt, async (err, hash) => {
req.body.password = hash;
db.collection('users').doc(email).set(req.body)
.then(async () => {
res.json({
name: req.body.name,
email: req.body.email,
seller: req.body.seller,
}); ...
```

Авторизація користувачів реалізована за допомогою маршрутів «/login». GET-запит відкриває сторінку входу, а POST-запит обробляє введені дані. Сервер отримує електронну адресу та пароль, після чого перевіряє, чи заповнені обидва поля. Якщо одне з полів порожнє, клієнтській частині повертається повідомлення про необхідність заповнення всіх полів.

3.1.3 Реалізація роботи з товарами

Робота з товарами реалізована за допомогою набору серверних маршрутів, які забезпечують додавання, отримання та видалення товарів. Для відкриття сторінки додавання товару використовується маршрут «/addProduct». Також передбачено маршрут «/addProduct/:id», який дозволяє відкривати сторінку додавання товару з урахуванням ідентифікатора. Це може використовуватися для роботи з уже створеним товаром.

Додавання товару виконується через POST-запит до маршруту «/addProduct». Сервер отримує дані товару, зокрема назву, короткий опис, повний опис, масив зображень, початкову ціну, розмір знижки, кінцеву ціну, кількість товару на складі, теги, електронну адресу продавця, ознаку чернетки та ідентифікатор товару.

Після успішної перевірки введених даних формується запит на створення посилання на документ у колекції товарів. Якщо у запиті передано ідентифікатор товару, серверна частина звертається до документа з відповідним ідентифікатором. Після виконання цих операцій сформований об’єкт товару зберігається у колекції товарів.

Лістинг 3.3 Реалізація додавання товару до бази даних:

```
let docRef = id
? db.collection(‘products’).doc(id)
: db.collection(‘products’).doc();
```

```
const productToSave = {
...req.body,
id: docRef.id,
draft: draft === true };
docRef.set(productToSave)
.then(() => {
res.json({ product: name }); })
.catch(() => {
res.json({ alert: 'Some Error Occured. Try Again' }); });
```

Отримання товарів реалізовано через POST-запит до маршруту «/get-products», який може працювати в кількох режимах залежно від переданих параметрів. Під час формування відповіді кожен отриманий документ перетворюється на об'єкт, після чого товар включається до масиву результатів і повертається клієнтській частині у форматі JSON.

Видалення товару реалізовано через POST-запит до маршруту «/delete-product»: сервер отримує ідентифікатор товару, знаходить відповідний документ у колекції товарів та видаляє його. Після успішного виконання операції клієнтська частина отримує відповідь «успішно», а у випадку помилки повертається відповідь «помилка».

Таким чином, серверна частина забезпечує основні операції для роботи з товарами, зокрема додавання нових позицій, збереження чернеток, отримання товарів за різними параметрами та видалення товарів із бази даних.

3.1.4 Реалізація оформлення замовлення

Оформлення замовлення реалізовано через POST-запит до маршруту «/create-order». Цей маршрут отримує від клієнтської частини об'єкт

замовлення, який містить електронну адресу користувача, перелік товарів, їх кількість, ціну, загальну суму та інші дані, необхідні для оформлення покупки. Отримані дані проходять попередню перевірку на коректність, після чого передаються на подальшу обробку серверною логікою.

Після отримання даних сервер формує HTML-представлення переліку товарів для електронного повідомлення, яке надсилатиметься користувачу після успішного оформлення замовлення. Для цього масив товарів обробляється за допомогою методу «map», у результаті чого для кожної товарної позиції створюється окремий рядок таблиці. У цьому рядку зазначається назва товару, кількість, ціна одиниці та загальна вартість позиції, що дозволяє користувачу отримати повну інформацію про склад свого замовлення в зручному вигляді.

Після формування даних замовлення сервер створює новий документ у колекції товарів. До документа додаються всі отримані дані, дата створення замовлення та початковий статус. Такий статус означає, що замовлення створено, але ще не перейшло до наступних етапів обробки.

Лістинг 3.4 Реалізація створення замовлення:

```
const orderRef = await db.collection('orders').add({
...orderData,
createdAt: new Date(),
status: "pending" });
```

Після збереження замовлення у базі даних сервер надсилає користувачу електронне повідомлення з підтвердженням, що забезпечує зворотний зв'язок та підвищує рівень довіри до платформи. У повідомленні зазначається, що замовлення успішно оформлено, а також подається короткий підсумок покупки. Зокрема, користувач отримує перелік товарів, їх кількість, ціну, вартість доставки, загальну суму та орієнтовний термін доставки.

3.2 Реалізація клієнтської частини вебзастосунку

3.2.1 Реалізація основних сторінок вебзастосунку

У вебзастосунку Urbix Store реалізовано набір сторінок, які відповідають основним сценаріям роботи користувача та продавця. Кожна сторінка забезпечує доступ до відповідного функціоналу залежно від ролі користувача в системі. Для звичайного користувача передбачено сторінки перегляду товарів, детального перегляду товару, реєстрації, входу, кошика та оформлення замовлення. Для користувача зі статусом продавця передбачено сторінку управління товарами та сторінку додавання або редагування товару.

Сторінка реєстрації та сторінка входу використовують спільну логіку обробки форми. Якщо на сторінці присутнє поле імені, форма працює в режимі реєстрації. У цьому випадку перевіряється довжина імені, наявність електронної пошти, довжина пароля, номер телефону та підтвердження умов використання. Якщо поле імені відсутнє, форма працює в режимі авторизації, де перевіряється заповнення електронної пошти та пароля. Після успішної перевірки дані передаються на сервер через функцію «sendData».

Лістинг 3.5 Реалізація обробки форми реєстрації та авторизації:

```
SubmitBtn.addEventListener('click', () => {
if (name !== null) {
if (name.value.length < 3) {
showAlert('Name Must Be 3 Letters Long');
} else if (!email.value.length) {
...} else {
loader.style.display = 'block';
sendData('/signup', {
name: name.value,
email: email.value,
...     seller: false }); }
```

Сторінка продавця реалізує відображення товарів, які належать конкретному користувачу. Для цього з локального сховища отримується інформація про поточного користувача, після чого на сервер передається його електронна адреса як ідентифікатор для фільтрації товарів у базі даних. У відповідь сервер повертає товари, створені цим продавцем. Якщо товари відсутні, на сторінці відображається відповідне зображення. Якщо товари є, кожен із них передається у функцію створення картки товару, яка формує представлення позиції з основною інформацією та кнопками для редагування і видалення.

Сторінка додавання товару реалізує два режими роботи: створення нового товару та редагування вже існуючого, що дозволяє уникнути дублювання інтерфейсу для подібних за призначенням операцій. Якщо ідентифікатор у адресі сторінки відсутній, форма відображається порожньою, що відповідає режиму створення нового товару. Такий підхід дозволяє продавцю редагувати створені товари без повторного введення інформації, а також спрощує підтримку клієнтської частини вебзастосунку.

Лістинг 3.6 Реалізація отримання даних товару для редагування:

```
const fetchProductData = () => {
 fetch('/get-products', {
  ...})
  .then(res => res.json())
  .then(data => {
   if (data.product) {
    setFormsData(data.product);
   } else {
    location.replace('/seller');
   }    })
  .catch(err => {
   location.replace('/seller'); });};
```

3.2.2 Реалізація взаємодії клієнтської частини з API

Взаємодія клієнтської частини з серверною реалізована за допомогою браузерного інтерфейсу запитів «fetch». Через нього виконуються запити для основних процесів вебзастосунку: реєстрації та авторизації користувачів, додавання й видалення товарів, подання заявки на статус продавця та оформлення замовлення.

Під час додавання товару клієнтська частина очікує завершення завантаження зображень, перевіряє форму та формує об’єкт товару. Якщо виконується редагування, до об’єкта додається ідентифікатор товару. Після цього дані надсилаються на маршрут «/addProduct». Також передбачено можливість збереження товару як чернетки через додавання поля «чернетка» зі значенням «істина».

Лістинг 3.7 Реалізація надсилання даних товару на сервер:

```
addProductBtn.addEventListener(‘click’, async () => {
await waitForUploads();
if (validateForm()) {
loader.style.display = ‘block’;
let data = productData();
if (productId) {
data.id = productId; }
sendData(‘/addProduct’, data); } });
saveDraft.addEventListener(‘click’, async () => {
await waitForUploads();
...
sendData(‘/addProduct’, data); } });
```

Для видалення товару реалізовано окремий запит до маршруту «/delete-product». Перед виконанням операції користувач підтверджує видалення,

після чого ідентифікатор товару передається на сервер. У разі успішної відповіді сторінка оновлюється, і товар більше не відображається у списку продавця.

Лістинг 3.8 Реалізація видалення товару:

```
const deleteItem = (id) => {
   fetch('/delete-product', {
      method: 'POST',
      headers: new Headers({ 'Content-Type': 'application/json' }),
      body: JSON.stringify({ id: id })
   }).then(res => res.json())
      .then(data => {
         if (data == 'success') {
            location.reload();
         } else {
            showAlert('Some error has occured while deleting the
product');}})}
```

Для подання заявки на статус продавця клієнтська частина надсилає дані форми на маршрут «/seller». До запиту додається електронна адреса поточного користувача, отримана з LocalStorage, що дозволяє серверній частині пов’язати заявку з конкретним обліковим записом.

Отже, взаємодія клієнтської частини з API забезпечує передавання даних між інтерфейсом користувача та серверною логікою для основних сценаріїв роботи вебзастосунку.

3.3 Реалізація системи ролей та прав доступу

У вебзастосунку Urbix Store реалізовано систему ролей, яка дозволяє розмежувати можливості покупця та продавця. Після реєстрації кожен

користувач отримує базову роль покупця, що дає змогу переглядати товари, додавати їх до кошика, користуватися списком бажаного та оформлювати замовлення. Для отримання функцій продавця користувач повинен заповнити відповідну форму реєстрації.

Роль користувача визначається за допомогою поля продавця, яке зберігається у колекції користувачів. Такий підхід дозволяє перевіряти роль користувача під час кожного запиту до серверної частини. Під час створення нового облікового запису значення цього поля встановлюється як «хибність», що відповідає ролі покупця. Після успішного подання заявки на статус продавця сервер створює документ із даними про магазин та оновлює поле на значення «істина». Це оновлення одразу відображається на клієнтській частині, надаючи користувачу доступ до сторінок управління товарами.

Лістинг 3.9 Реалізація надання користувачу статусу продавця:

```
app.post('/seller', (req, res) => {
  let { name, about, address, number, tac, legit, email } = req.body;
  if (!name.length || !address.length || !about.length || number.length < 10 ||
    !Number(number)) {
    return res.json({ 'alert': 'Some Information is Invalid' });
  } else if (!tac || !legit) {
    return res.json({ 'alert': 'You Must Agree to Our Terms and
Conditions' });
  } else {
    db.collection('sellers').doc(email).set(req.body)
      .then(data => {
        db.collection('users').doc(email).update({
          seller: true
        }).then(data => {
          res.json(true);
      ... });
```

На клієнтській стороні інформація про користувача зберігається у LocalStorage. Це дозволяє перевіряти факт авторизації та визначати, чи має користувач доступ. Після оновлення статусу користувачу стають доступними функції додавання, редагування, видалення та публікації товарів.

3.4 Реалізація роботи з базою даних

Робота з базою даних у вебзастосунку Urbix Store реалізована через серверну частину з використанням Firebase Firestore. У системі застосовуються колекції користувачів, продавців, товарів та замовлень, які були спроєктовані відповідно до основних сутностей вебзастосунку.

Оскільки підключення до Firebase Firestore було налаштовано на етапі конфігурації серверного середовища, подальша робота з базою даних здійснюється через змінну «db».

Під час реєстрації користувача сервер створює документ у колекції користувачів. Ідентифікатором документа є електронна адреса користувача, що дозволяє перевіряти наявність облікового запису під час повторної авторизації. Приклад збереженого документа користувача у Firebase Firestore наведено на рисунку 3.1.

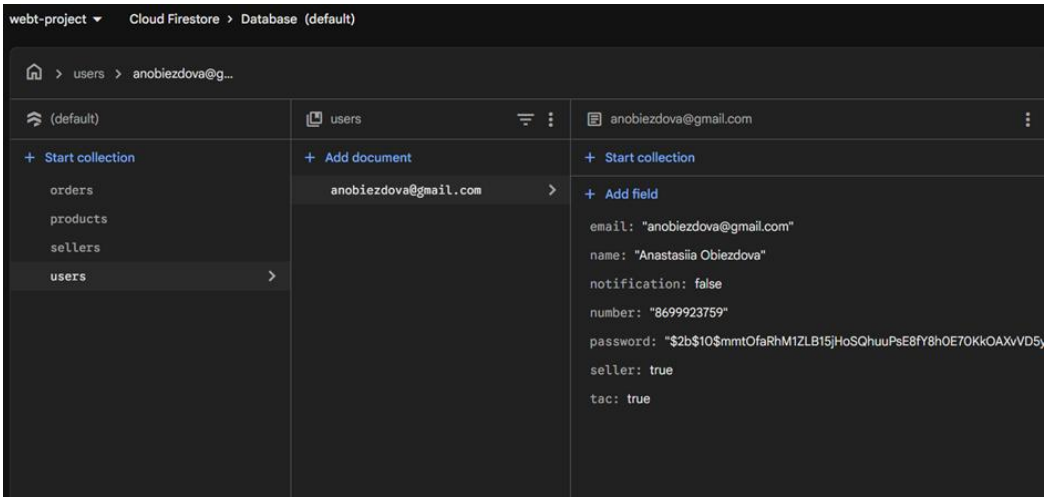


Рисунок 3.1 – Приклад документа користувача в колекції користувачів Firebase Firestore

Отримання даних з бази також реалізовано через серверну частину. Наприклад, під час відкриття сторінки товару клієнтська частина передає ідентифікатор товару, після чого сервер отримує відповідний документ із колекції товарів. Для відображення товарів конкретного продавця сервер виконує запит за електронною адресою продавця, а для формування каталогу повертає документи, які не мають ознаки чернетки.

Збереження товарів реалізовано через колекцію товарів. Якщо продавець додає новий товар, сервер створює новий документ у цій колекції. Якщо виконується редагування вже існуючого товару, використовується переданий ідентифікатор документа. До об’єкта товару додається його ідентифікатор, а також значення поля чернетки, яке визначає, чи є товар чернеткою.

Лістинг 3.10 Реалізація збереження товару в колекції товарів:

```
let docRef = id
   ? db.collection('products').doc(id)
   : db.collection('products').doc();
const productToSave = {
   ...req.body,
   id: docRef.id,
   draft: draft === true
};

docRef.set(productToSave)
   .then(() => {
      res.json({ product: name });
   });
```

Під час оформлення замовлення сервер створює новий документ у колекції замовлень. До нього додаються дані замовлення, дата створення та

початковий статус. Такий підхід дозволяє зберігати історію оформлених покупок і надалі розширювати функціонал, наприклад додавати зміну статусів замовлення.

Лістинг 3.11 Реалізація створення документа замовлення:

```
const orderRef = await db.collection('orders').add({
   ...orderData,
   createdAt: new Date(),
   status: "pending"
});
```

Таким чином, робота з базою даних у вебзастосунку Urbix Store охоплює створення користувачів, реєстрацію продавців, збереження товарів, отримання даних для відображення на сторінках та створення замовлень. Усі операції виконуються через серверну частину, що дозволяє централізовано контролювати взаємодію з Firestore та забезпечувати узгодженість даних між клієнтською частиною і базою даних.

3.5 Інтеграція із зовнішніми сервісами

У процесі розроблення вебзастосунку Urbix Store було реалізовано інтеграцію з рядом зовнішніх сервісів. Використання сторонніх сервісів забезпечує оптимізацію обробки даних, зменшення навантаження на сервер та покращення користувацького досвіду.

Взаємодія із зовнішніми сервісами реалізована із використанням змінних середовища. Для цього у проєкті застосовано файл «.env», у якому зберігаються конфіденційні дані, зокрема ключі доступу до Firebase та Amazon Web Services. Такий підхід дозволяє забезпечити безпеку даних та уникнути їх збереження безпосередньо у програмному коді.

Для збереження зображень товарів реалізовано інтеграцію з Amazon S3 (рис. 3.2). У системі використано механізм генерації тимчасових URL-адрес, через які клієнтська частина безпосередньо завантажує файли у хмарне сховище.

Лістинг 3.12 Приклад реалізації генерації URL:

```
const params = {
    Bucket: bucketName,
    Key: imageName,
    Expires: 300,
    ContentType: 'image/jpeg'
};
const uploadUrl = await s3.getSignedUrlPromise('putObject', params);
```

Після отримання такого URL клієнтська частина виконує пряме завантаження файлу у S3, що дозволяє уникнути передачі файлів через сервер та зменшує навантаження на серверну частину застосунку. Такий підхід є ефективним з точки зору продуктивності, оскільки медіафайли передаються безпосередньо між браузером та хмарним сховищем.

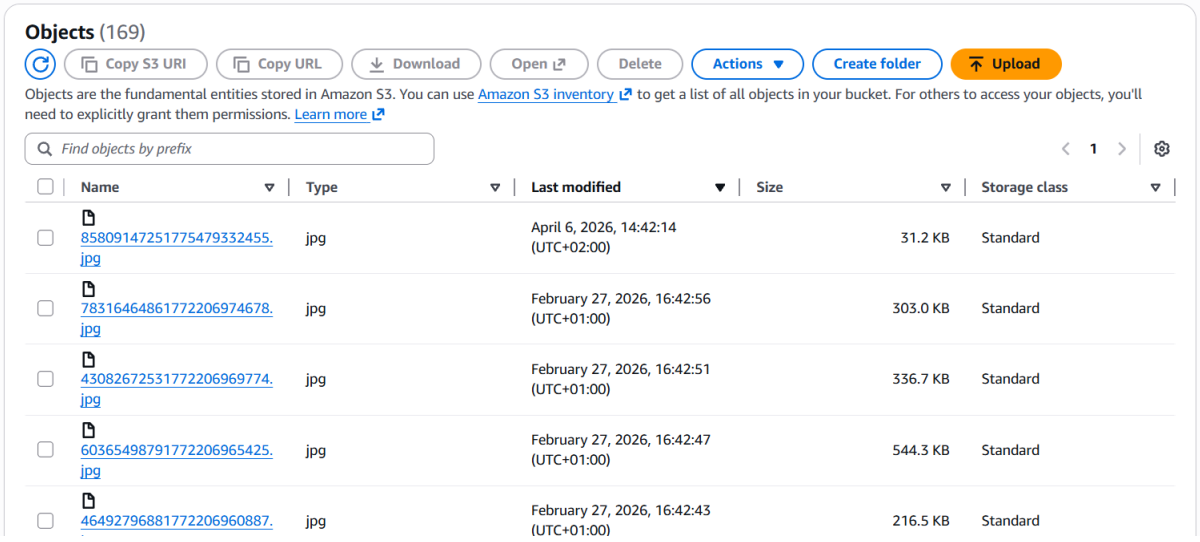


Рисунок 3.2 – Збереження зображень товарів у хмарному сховищі Amazon S3

Для автоматичного інформування користувачів реалізовано два типи електронних повідомлень: повідомлення після реєстрації та підтвердження оформленого замовлення.

Лістинг 3.13 Приклад надсилання електронного повідомлення:

```
await transporter.sendMail({
  from: `"Urbix Store" <${process.env.EMAIL_USER}>`,
  to: email,
  subject: "Welcome to Urbix",
  html: `<h2>Welcome, ${name}!</h2>`
});
```

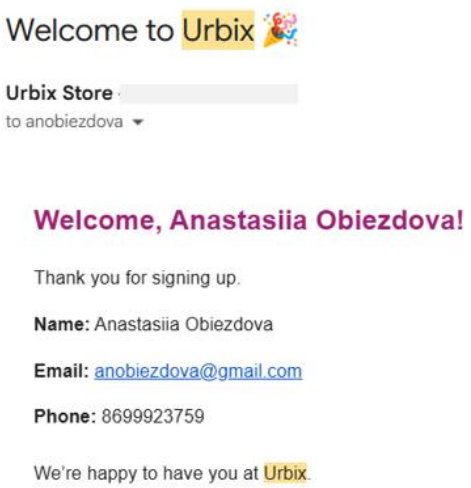
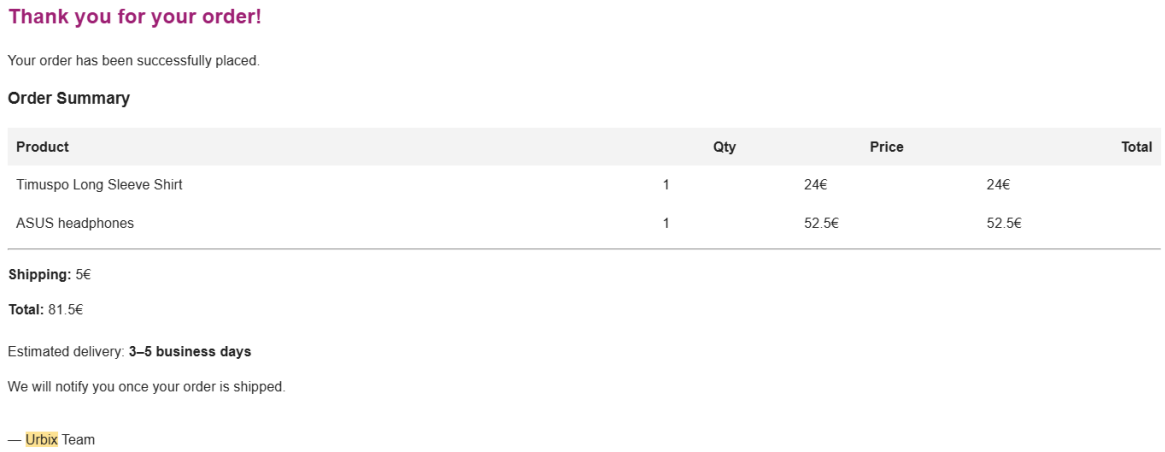


Рисунок 3.3 – Приклад електронного повідомлення після реєстрації користувача

Також у системі реалізовано надсилання листа після оформлення замовлення, який містить інформацію про товари, їх кількість та загальну вартість (рис. 3.4).

Наведені повідомлення містять основну інформацію про виконану дію та забезпечують зворотний зв'язок із користувачем, що підвищує зручність використання системи.

Слід зазначити, що у даній реалізації не було інтегровано реальні платіжні системи. Натомість процес вибору способу оплати реалізовано на рівні інтерфейсу користувача, що дозволяє імітувати оформлення замовлення без виконання фінансових операцій.



Thank you for your order!

Your order has been successfully placed.

Order Summary

Product	Qty	Price	Total
Timuspo Long Sleeve Shirt	1	24€	24€
ASUS headphones	1	52.5€	52.5€

Shipping: 5€

Total: 81.5€

Estimated delivery: **3–5 business days**

We will notify you once your order is shipped.

— Urbix Team

Рисунок 3.4 – Приклад електронного повідомлення після оформлення замовлення

Таким чином, інтеграція з зовнішніми сервісами дозволила реалізувати зберігання медіафайлів, обробку даних та систему сповіщень, що є важливими складовими сучасних вебзастосунків електронної комерції

3.6 Демонстрація роботи вебзастосунку

3.6.1 Демонстрація роботи користувацької частини

Інтерфейс вебзастосунку Urbix Store реалізовано у сучасному мінімалістичному стилі, що забезпечує чисте та зрозуміле візуальне середовище без зайвих елементів. Дизайн орієнтований на молоду аудиторію та користувачів, які цікавляться трендовими товарами, тому кольорова гама та типографіка підібрані відповідно до сучасних тенденцій у веб-дизайні. Інтерфейс забезпечує швидкий доступ до основних функцій системи: пошуку

товарів, перегляду категорій, додавання товарів у кошик та оформлення замовлення. Навігація побудована таким чином, щоб користувач міг інтуїтивно переходити між розділами, не витрачаючи часу на пошук потрібних елементів керування.

Першим етапом взаємодії користувача із системою є перегляд головної сторінки вебзастосунку. Головна сторінка є центральною точкою доступу до основних функцій системи та формує перше враження користувача про платформу, тому її структура спроєктована з урахуванням принципів зручності та логічності розміщення елементів. У верхній частині сторінки розміщено навігаційну панель, яка містить логотип, рядок пошуку, іконки профілю користувача та кошика, а також посилання для швидкого переходу між основними розділами вебзастосунку (рис. 3.5). Навігаційна панель є фіксованою і залишається видимою під час прокручування сторінки, що забезпечує постійний доступ до ключових функцій платформи незалежно від позиції користувача на сторінці.

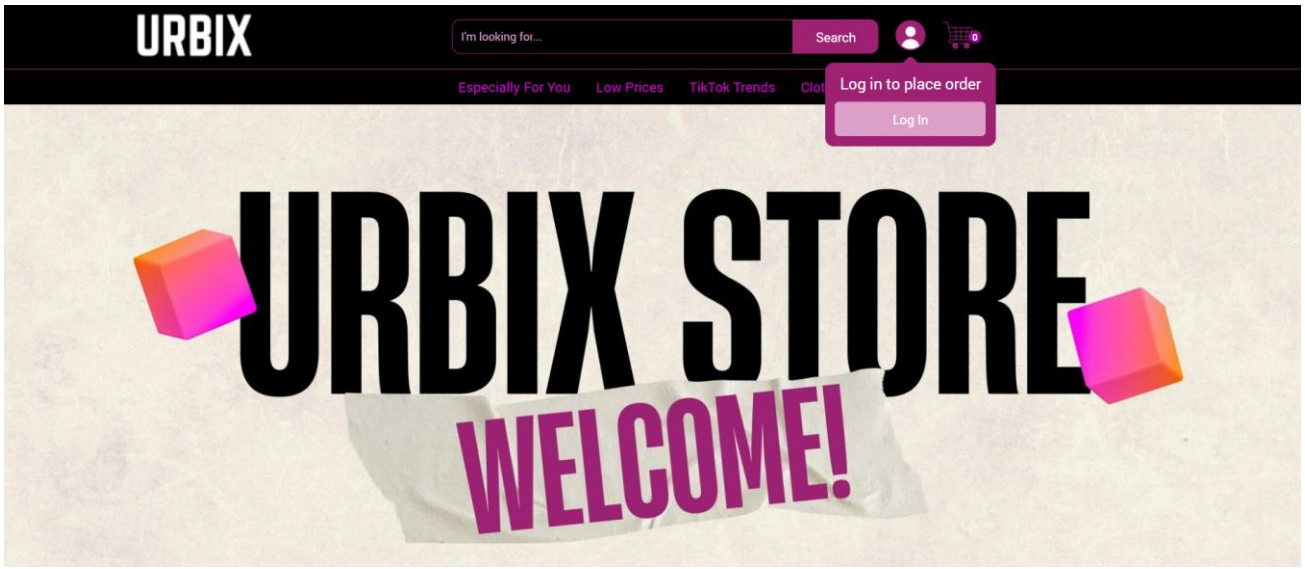


Рисунок 3.5 – Початок головної сторінки вебзастосунку та навігаціона панель

На головній сторінці представлено блоки товарів, згруповані за категоріями, такими як «Especially For You», «Low Prices», «TikTok Trends»

та «Clothes» (рис. 3.6). Кожен товар відображається у вигляді картки, що містить зображення товару, назву, короткий опис, поточну ціну та розмір знижки.

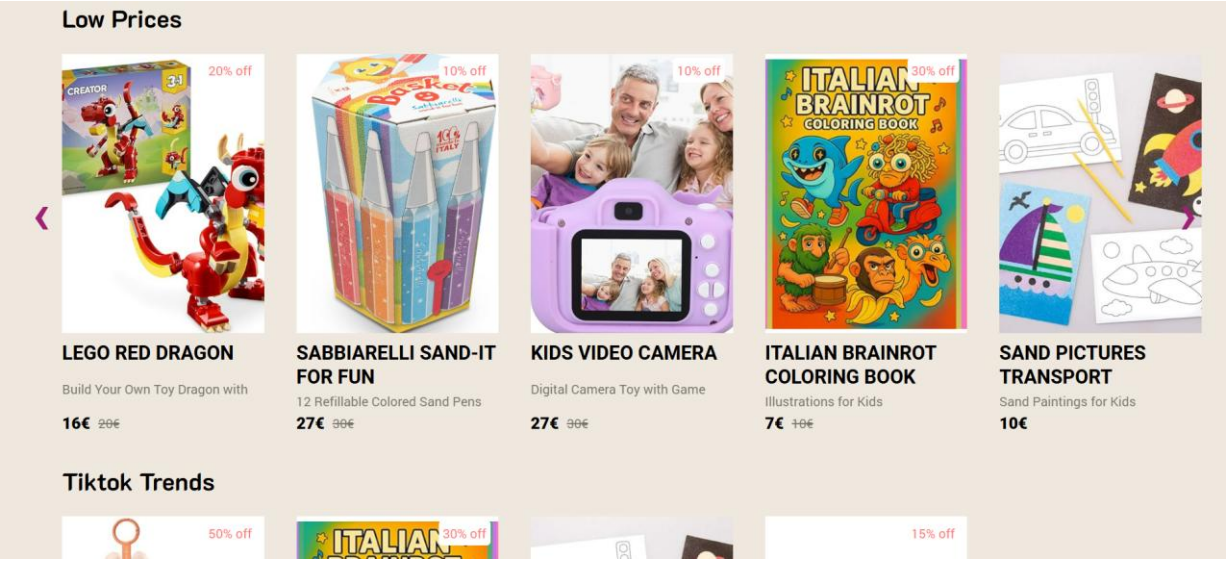


Рисунок 3.6 – Головна сторінка вебзастосунку

Після вибору товару користувач переходить на сторінку детального перегляду продукту (рис 3.7). Сторінка товару призначена для ознайомлення користувача з основною інформацією про обраний продукт. У лівій частині сторінки розміщено основне зображення товару та галерею мініатюр, що дозволяє переглядати товар з різних ракурсів. У правій частині відображено назву товару, короткий опис, ціну та розмір знижки. Така структура дозволяє користувачу бачити візуальне представлення товару та його характеристики.

Користувач має можливість додати товар до кошика за допомогою кнопки «Add to cart» або зберегти його до списку бажаного за допомогою кнопки «Add to wishlist». Після додавання товару до кошика інтерфейс оновлюється, а кількість товарів у кошику змінюється в режимі реального часу. Таким чином, користувач завжди бачить актуальний стан свого кошика без необхідності переходити на окрему сторінку. У нижній частині сторінки розміщено розширений опис товару, який містить інформацію про матеріали та характеристики.

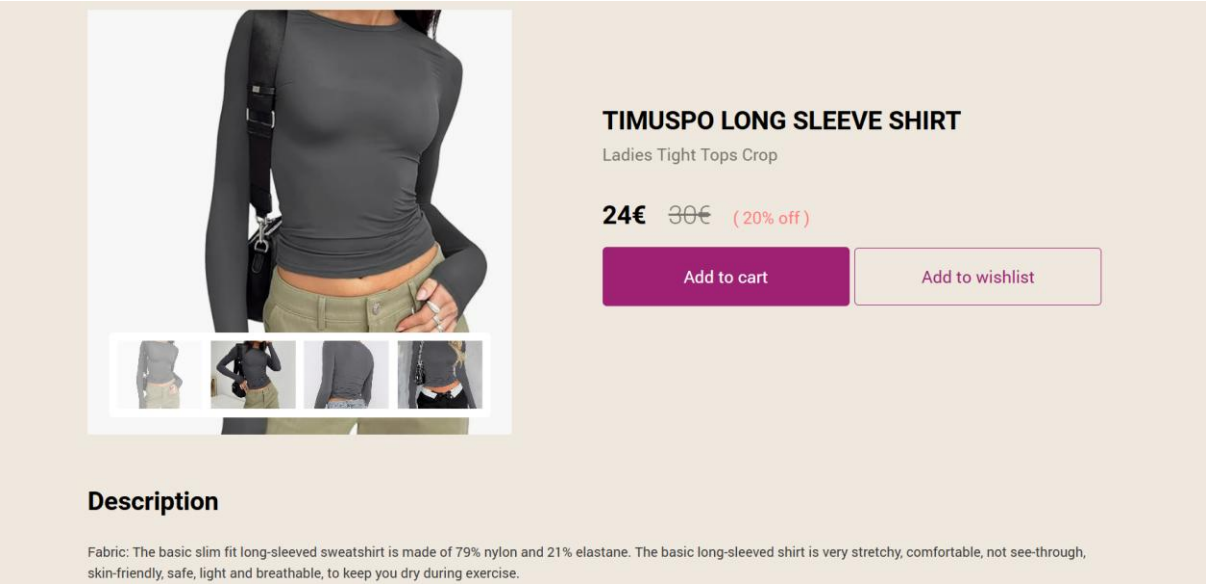


Рисунок 3.7 – Сторінка детального перегляду товару

У нижній частині сторінки товару розміщено блок схожих товарів, який призначений для підвищення зацікавленості користувача та переходу до перегляду інших товарів платформи (рис. 3.8). Рекомендовані товари відображаються у вигляді горизонтального списку. Кожна картка містить зображення товару, назву, короткий опис, поточну та попередню ціну, а також відсоток знижки за наявності.

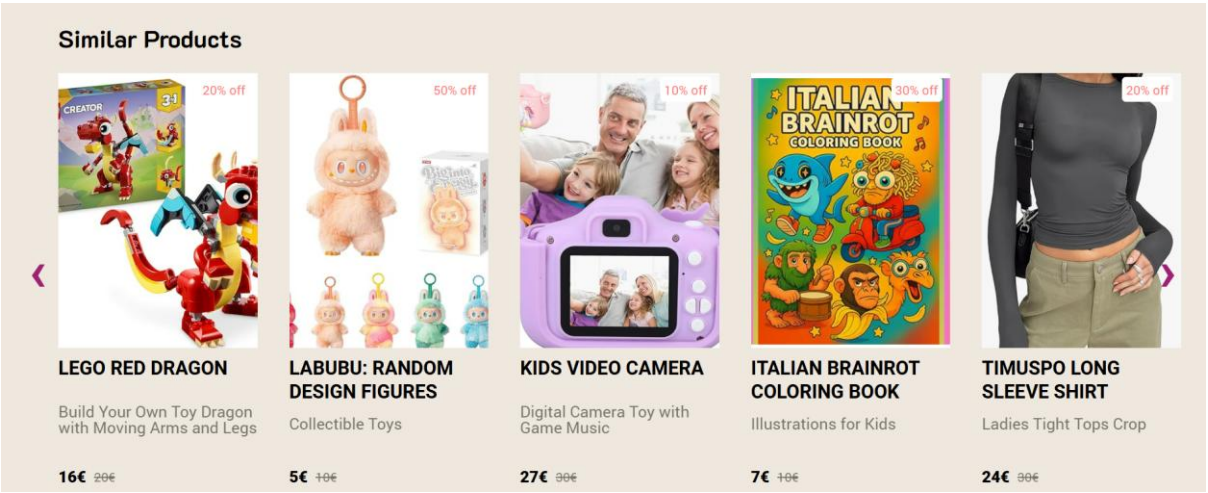


Рисунок 3.8 – Блок схожих товарів на сторінці товару

У нижній частині вебзастосунку розміщено footer-блок, який містить інформацію про платформу, посилання на категорії товарів та контактні дані.

Також у цьому блоці реалізовано кнопку «Apply Here», за допомогою якої користувач може перейти до подання заявки на отримання статусу продавця (рис. 3.9).

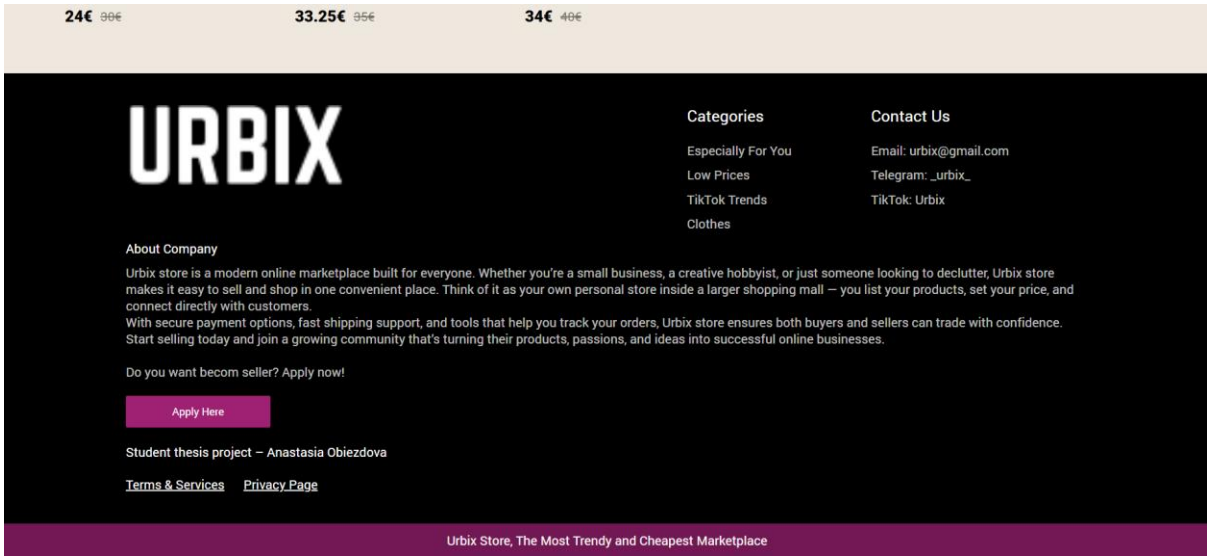


Рисунок 3.9 – Нижня частина вебзастосунку

Для оформлення замовлення користувач повинен бути авторизованим у системі. Сторінка входу призначена для авторизації зареєстрованих користувачів (рис. 3.10). Перехід до неї здійснюється шляхом натискання на елемент «Log In» у навігаційній панелі. Після натискання кнопки «Log In» система перевіряє введені дані. У разі успішної авторизації користувач отримує доступ до функціоналу оформлення замовлення, а у випадку помилки відображається відповідне повідомлення.

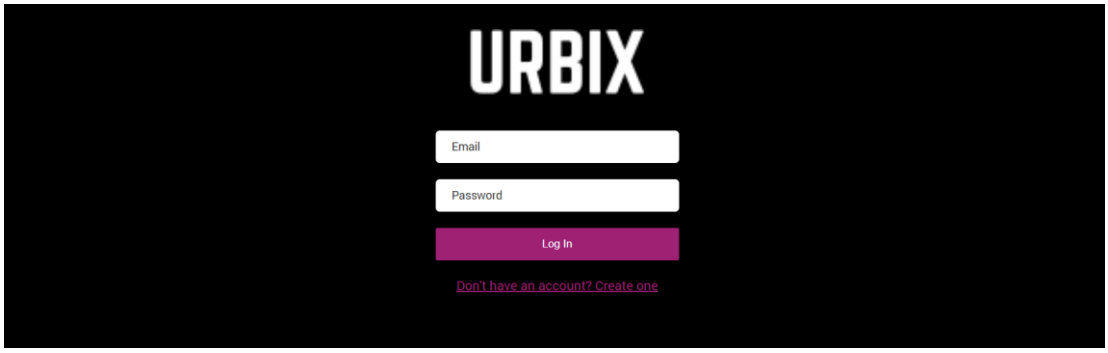


Рисунок 3.10 – Сторінка входу в систему

Якщо користувач не має облікового запису, він може перейти до сторінки реєстрації за допомогою посилання. Сторінка реєстрації нового облікового запису призначена для створення профілю користувача в системі (рис. 3.11). Форма передбачає введення імені, електронної пошти, пароля та номера телефону.

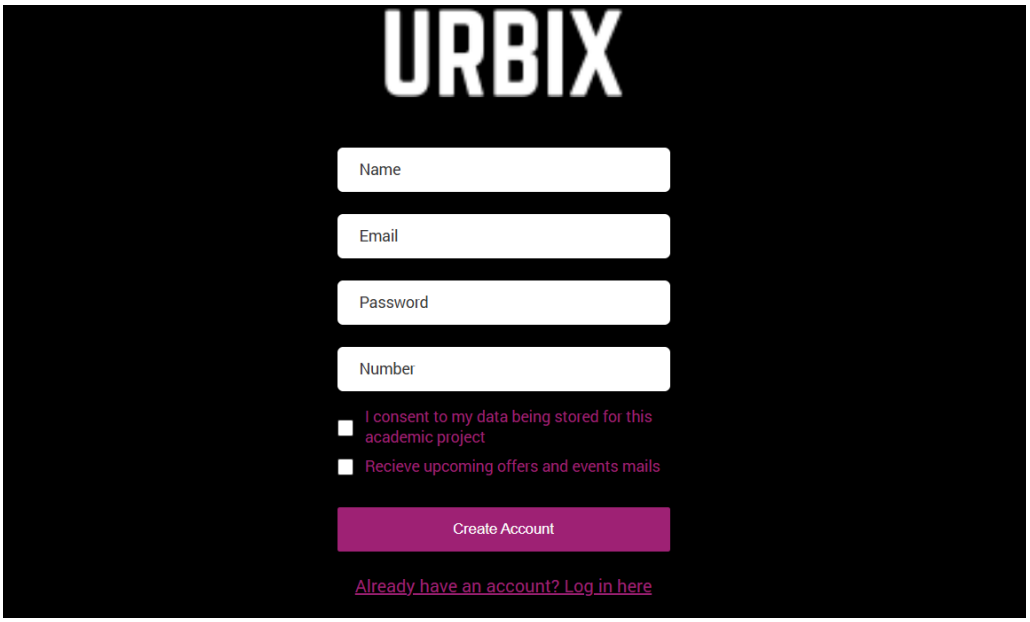


Рисунок 3.11 – Сторінка реєстрації користувача

Після натискання кнопки «Create Account» система виконує перевірку введених даних і створює новий обліковий запис. Також перевіряється наявність введеної електронної пошти в базі даних. Якщо така пошта вже зареєстрована, користувачу відображається відповідне повідомлення.

Після додавання товарів користувач може перейти до кошика, натиснувши на відповідну іконку в навігаційній панелі. Сторінка кошика відображає перелік обраних користувачем товарів (рис. 3.12). Кожен товар подано у вигляді окремого рядка, що містить зображення, назву, короткий опис, ціну та елемент керування кількістю товару. Користувач має можливість змінювати кількість товарів за допомогою кнопок «+» і «-», видаляти товар із кошика та переглядати вартість кожної позиції.

У правій частині сторінки відображено підсумкову вартість замовлення та кнопку «Checkout», яка ініціює процес оформлення покупки. Також реалізовано блок «Wishlist», у якому користувач може переглядати збережені товари.

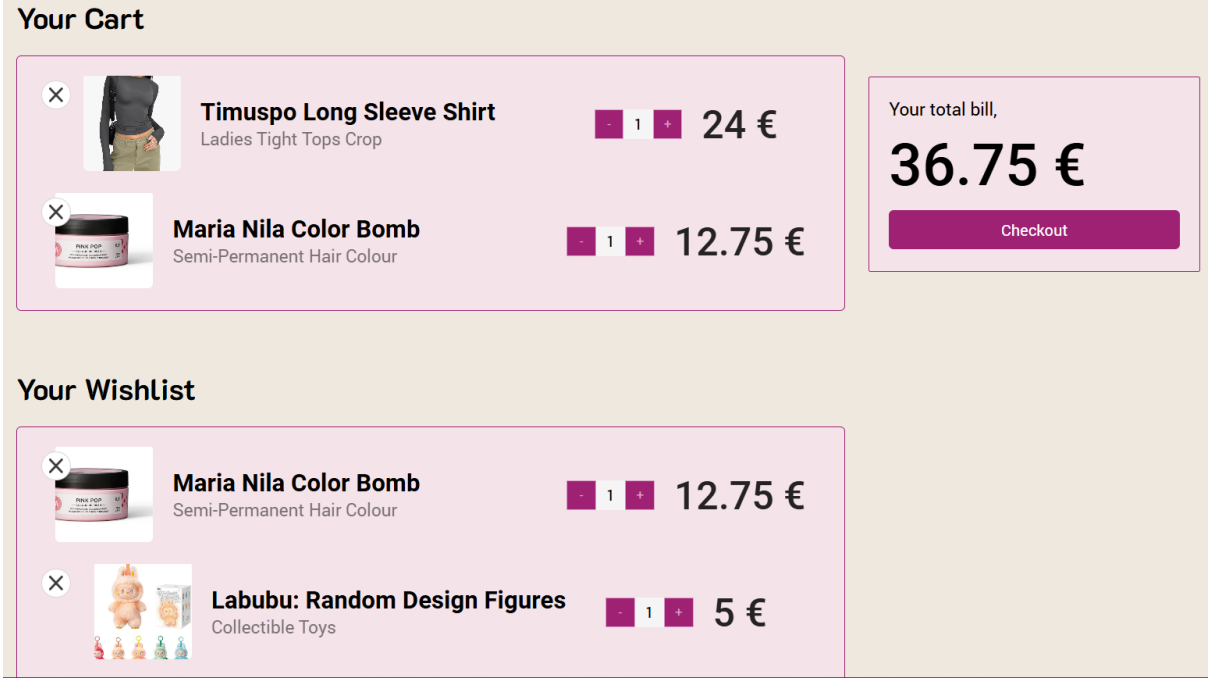


Рисунок 3.12 – Сторінка кошика

Після натискання кнопки «Checkout», за умови успішної авторизації користувача, відбувається перехід на сторінку оформлення замовлення (рис. 3.13). Форма оформлення замовлення передбачає введення імені отримувача, адреси, міста, штату, поштового індексу, номера телефону та підтвердження згоди на обробку персональних даних.

У правій частині сторінки розміщено блок «Order Summary», який відображає деталізовану інформацію про вартість замовлення: проміжну суму вартості обраних товарів, вартість доставки залежно від обраного способу, а також загальну суму до оплати, що формується автоматично на основі введених даних. Такий блок дозволяє користувачу перевірити коректність сформованого замовлення перед його підтвердженням та уникнути помилок, пов’язаних із неправильним розрахунком вартості.

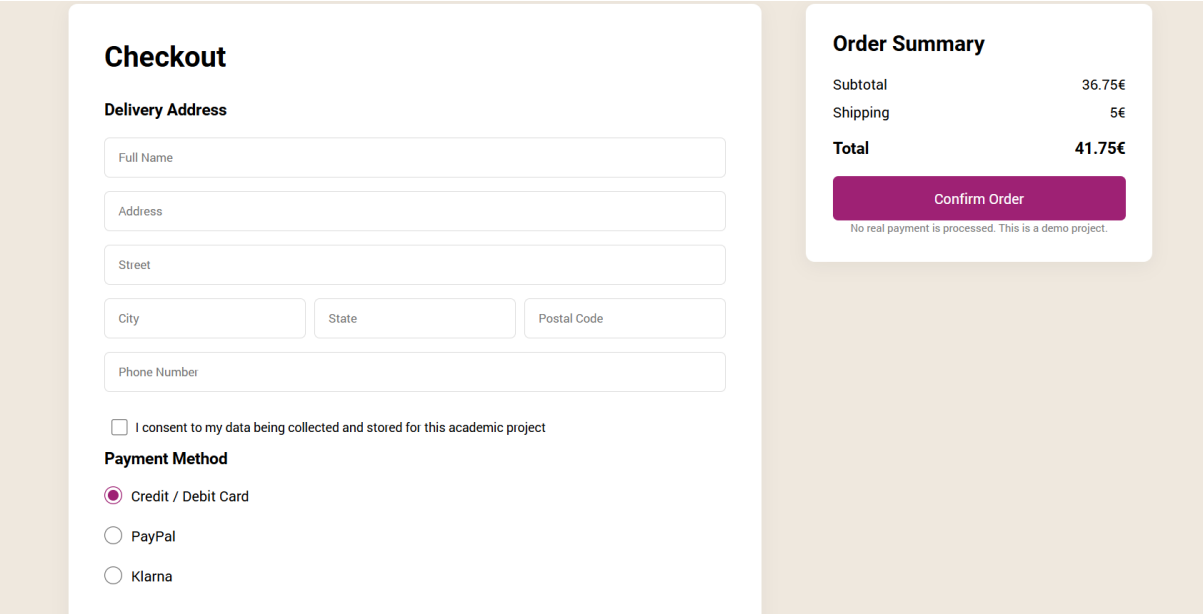


Рисунок 3.13 – Сторінка оформлення замовлення

Після заповнення всіх обов'язкових полів форми та перевірки введених даних користувач натискає кнопку «Confirm Order», після чого клієнтська частина надсилає запит на серверну частину для збереження замовлення. У разі успішного виконання запиту система завершує процес оформлення замовлення та відображає сторінку підтвердження (рис. 3.14).

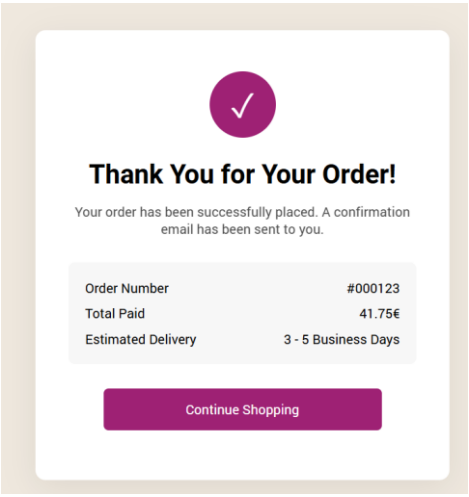


Рисунок 3.14 – Сторінка підтвердження замовлення

На сторінці підтвердження замовлення користувачу відображається номер замовлення, який присвоюється під час його створення у базі даних,

загальна сума оплати, що відповідає підсумковому значенню, розрахованому на попередньому етапі, а також орієнтовний термін доставки, який повідомляє користувачу про очікувані часові межі отримання товару. Така інформація дає користувачу підтвердження успішного завершення процесу покупки та формує уявлення про подальші кроки. Крім того, на сторінці реалізовано кнопку «Continue Shopping», натискання якої забезпечує повернення до перегляду товарів і дозволяє користувачу продовжити роботу з каталогом без необхідності повторного переходу через головну сторінку.

3.6.2 Демонстрація роботи продавця в системі

Окремим сценарієм роботи вебзастосунку є взаємодія користувача із системою в ролі продавця. Для отримання статусу продавця користувач натискає кнопку «Apply Here», розміщену у footer-блоці вебзастосунку. Після цього система перенаправляє його на сторінку подання заявки на отримання статусу продавця (рис. 3.15). На цій сторінці розміщено короткий опис умов співпраці, зокрема інформацію про правила платформи та комісію з продажів.

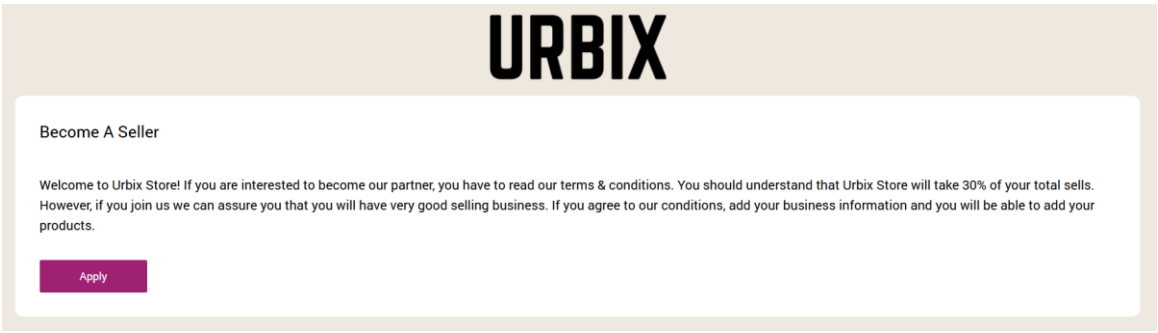


Рисунок 3.15 – Сторінка подання заявки на отримання статусу продавця

Після натискання кнопки «Apply» користувач переходить до форми реєстрації продавця (рис. 3.16). Форма передбачає введення назви магазину,

адреси, опису діяльності та контактного номера телефону. Також вона містить прапорці для підтвердження ознайомлення з умовами проєкту та надання згоди на збереження даних у базі. Після заповнення форми та натискання кнопки «Apply» система обробляє введені дані та створює профіль продавця.

Рисунок 3.16 – Форма реєстрації продавця

Після створення профілю продавця користувач отримує доступ до сторінки управління товарами (рис. 3.17). На цій сторінці відображається список доданих товарів у вигляді карток. Продавець має можливість переглядати власні товари, редагувати інформацію про них, видаляти позиції, додавати нові товари за допомогою кнопки «Add Product», а також переходити до перегляду магазину через кнопку «View Store».

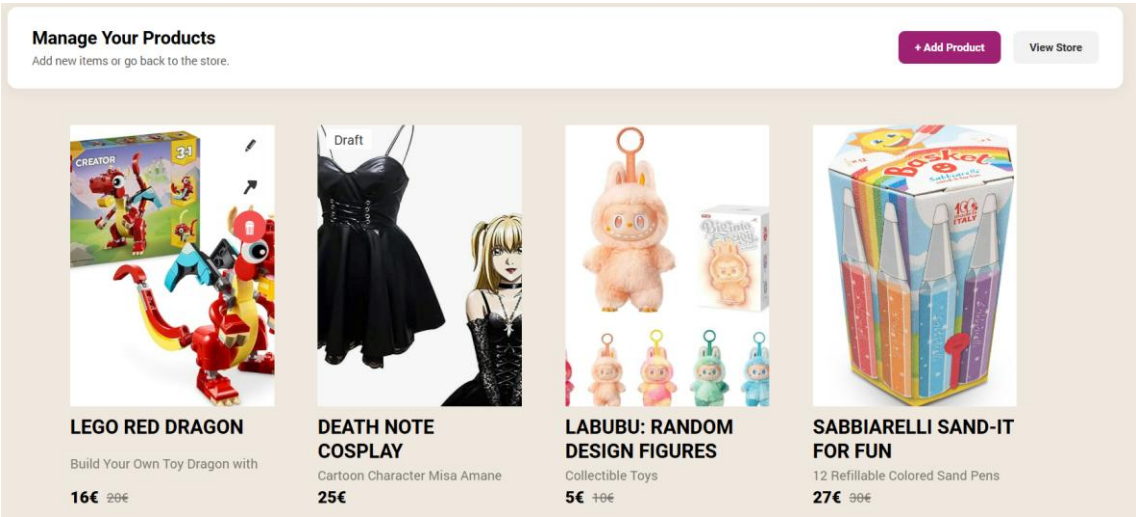
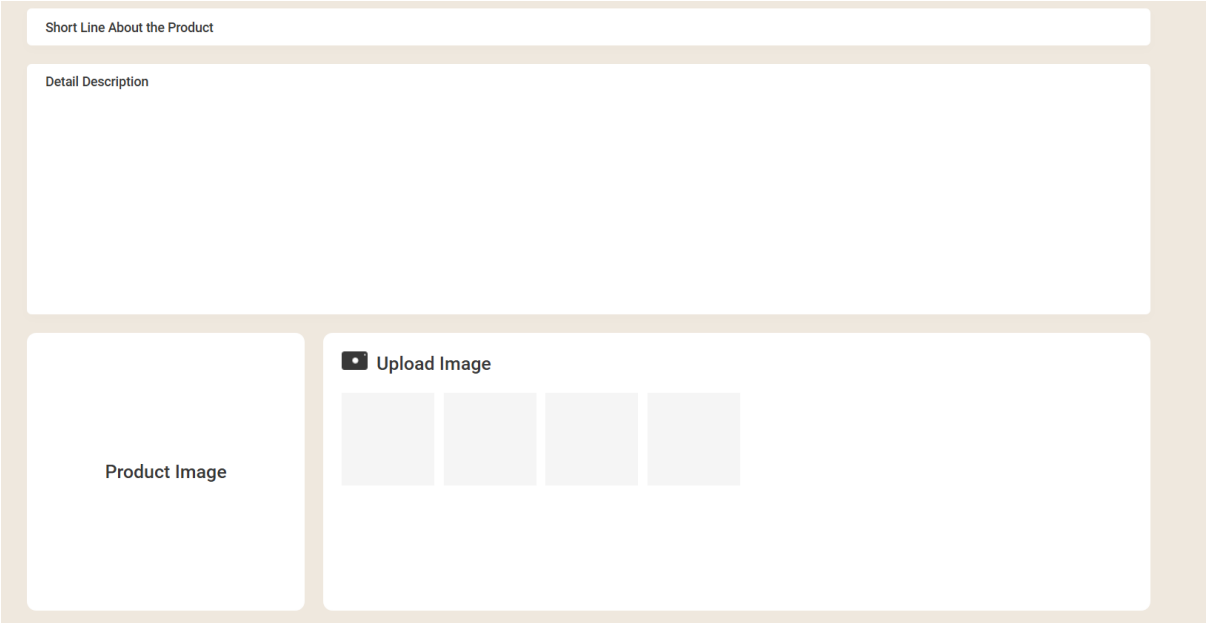


Рисунок 3.17 – Сторінка управління товарами продавця

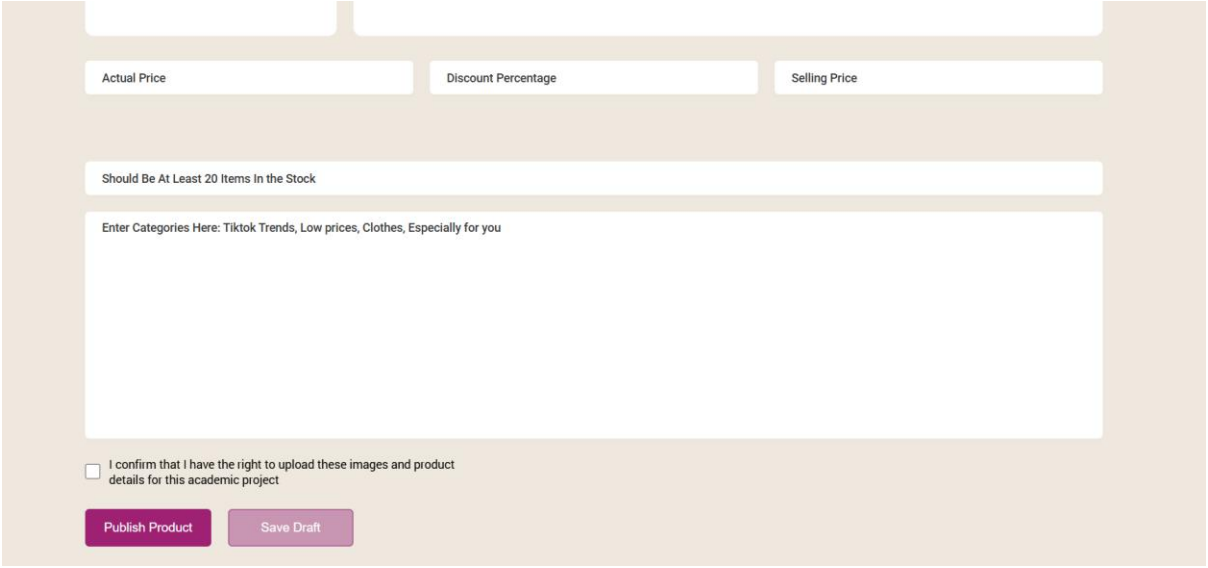
Якщо продавець натискає кнопку «Add Product» на сторінці управління товарами, система перенаправляє його на сторінку додавання нового товару (рис. 3.18), яка реалізована як форма з набором полів для введення основної інформації про товар. На цій сторінці продавець має можливість ввести назву товару, короткий опис, що використовується для відображення товару в каталозі, а також повний опис, який надає докладнішу інформацію на сторінці перегляду товару. Окрім текстових та графічних даних, продавець вказує повну ціну товару, розмір знижки за потреби, а також кількість товару на складі, що дозволяє системі відстежувати наявність товару та коректно відображати інформацію про доступність позиції. Додатково передбачено вибір категорії, у якій буде розміщено товар, що забезпечує його відображення у відповідному розділі каталогу.

Остаточна ціна товару розраховується автоматично з урахуванням зазначеної знижки, що зменшує ризик помилок під час встановлення ціни та звільняє продавця від необхідності вручну обчислювати кінцеву вартість. Після заповнення форми та натискання кнопки «Publish Product» виконується перевірка коректності введених даних, після чого товар додається до бази даних і стає доступним для відображення на платформі. У разі публікації товару він одразу з’являється в каталозі та може бути знайдений покупцями за допомогою пошуку або перегляду відповідної категорії. Якщо продавець не готовий до публікації, він може зберегти товар як чернетку та повернутися до його редагування пізніше.

Після реєстрації користувача як продавця навігаційна панель вебзастосунку змінюється, відображаючи розширений набір елементів керування, доступних лише для цієї ролі. У ній з’являється додаткова кнопка «Manage Products», яка забезпечує перехід до сторінки управління товарами продавця. Крім того, ім’я користувача відображається як назва зареєстрованого магазину, що дозволяє продавцю одразу ідентифікувати свій обліковий запис у системі (рис. 3.19).



а)



б)

Рисунок 3.18 – Сторінка додавання нового товару: а) початок форми додавання товару; б) завершальна частина форми додавання товару

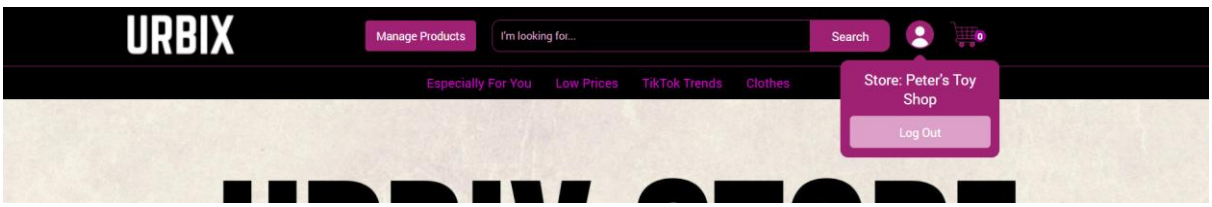


Рисунок 3.19 – Змінена навігаційна панель для користувача зі статусом продавця

3.7 Тестування вебзастосунку

У процесі розроблення вебзастосунку Urbix Store було проведено функціональне тестування, метою якого є перевірка коректності роботи основних компонентів системи та відповідності реалізованого функціоналу поставленим вимогам. Такий вид тестування дозволяє виявити можливі недоліки реалізації на ранніх етапах та підтвердити готовність системи до подальшого використання.

Тестування здійснювалося у ручному режимі шляхом перевірки основних сценаріїв взаємодії користувача з системою. Особливу увагу було приділено перевірці валідації введених даних, обробці помилок, а також взаємодії клієнтської частини з серверним API. Такий підхід дозволив переконатися, що система коректно реагує як на правильно заповнені форми, так і на введення некоректних або неповних даних.

Зокрема, було протестовано механізми валідації форм, реалізовані на клієнтській частині. Також перевіряється заповнення обов’язкових полів та вибір способу оплати. У випадку порушення умов валідації система не дозволяє продовжити процес оформлення замовлення, а користувачу відображається відповідне повідомлення про необхідність виправити введені дані.

Аналогічні перевірки реалізовано під час додавання товару продавцем. Ці перевірки забезпечують коректність введених даних та запобігають збереженню некоректної інформації у системі, а також підвищують надійність роботи вебзастосунку в цілому.

Тестування охоплювало як позитивні, так і негативні сценарії використання системи, що дозволило перевірити поведінку системи у штатних умовах роботи та у випадках некоректних дій користувача. Для наочності результати тестування основних функціональних можливостей було систематизовано у вигляді таблиці 3.1.

Таблиця 3.1 – Результати функціонального тестування вебзастосунку

№	Функція	Вхідні дані	Очікуваний результат	Фактичний результат
1	Реєстрація	Валідні дані	Створення користувача	Виконано
2	Реєстрація	Короткий пароль	Відображення помилки	Виконано
3	Логін	Правильні дані	Успішний вхід	Виконано
4	Логін	Невірний пароль	Повідомлення про помилку	Виконано
5	Додавання в кошик	Натискання кнопки	Товар додається у кошик	Виконано
6	Оформлення замовлення	Неавторизований користувач	Відображення повідомлення	Виконано
7	Оформлення замовлення	Невалідні дані	Блокування оформлення	Виконано
8	Оформлення замовлення	Валідні дані	Створення замовлення	Виконано
9	Продавець	Валідна форма	Отримання статусу продавця	Виконано
10	Додавання товару	Без зображення	Повідомлення про помилку	Виконано
11	Додавання товару	Валідні дані	Товар додається	Виконано
12	Завантаження зображення	Некоректний файл	Повідомлення про помилку	Виконано
13	Завантаження зображення	Зображення	Успішне завантаження AWS	Виконано
14	Створення замовлення	Валідні дані	Запис у БД + пошта	Виконано

У результаті проведеного тестування встановлено, що всі основні функціональні можливості вебзастосунку працюють коректно. Реалізовані механізми валідації дозволяють запобігати введенню некоректних даних, а інтеграція з серверною частиною забезпечує стабільну обробку запитів.

3.8 Аналіз результатів та перспективи подальшого розвитку

У результаті виконання кваліфікаційної роботи було розроблено клієнт-серверний вебзастосунок Urbix Store, який реалізує основні

функціональні можливості систем електронної комерції. Проведене тестування підтвердило коректність роботи ключових компонентів системи, а також стабільність взаємодії між клієнтською та серверною частинами.

У межах реалізації розроблено клієнт-серверну архітектуру вебзастосунку, реалізовано реєстрацію та авторизацію користувачів, перегляд каталогу товарів, кошик та оформлення замовлення, систему ролей, управління товарами продавця, а також інтеграцію із зовнішніми сервісами для зберігання зображень і надсилання повідомлень.

Разом з тим система має певні обмеження: відсутність інтеграції з платіжними системами, адміністративної панелі, системи відгуків і рейтингу, механізму відновлення пароля, а також обмежений функціонал пошуку та недостатній рівень оптимізації для мобільних пристроїв.

Одним із перспективних напрямів розвитку є впровадження інтелектуальних механізмів аналізу поведінки користувачів, зокрема систем кластеризації даних про перегляди та покупки для формування персоналізованих рекомендацій товарів. Сучасні методи інтелектуального аналізу та оброблення даних забезпечують ефективну основу для реалізації таких механізмів [34], зокрема через застосування систем формування ознакових просторів для класифікації даних [35]. Подібні підходи на основі ансамблевих методів кластеризації потоків даних дозволяють опрацьовувати інформацію про користувачів у режимі реального часу та адаптуватися до змін їхньої поведінки [36]. Інтеграція таких механізмів у вебзастосунок Urbix Store могла би підвищити релевантність пропонованих товарів та покращити загальний користувацький досвід.

Незважаючи на зазначені обмеження, розроблений вебзастосунок демонструє повноцінну реалізацію основних бізнес-процесів електронної комерції та може бути використаний як основа для подальшого розвитку, напрями якого визначаються виявленими обмеженнями.

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено і реалізовано клієнт-серверний вебзастосунок електронної комерції Urbix Store, який забезпечує взаємодію між покупцями та продавцями в онлайн-середовищі. Для досягнення поставленої мети вирішено такі завдання:

– проведено аналіз предметної області електронної комерції та розглянуто існуючі платформи Amazon, OLX та Prom, що дало можливість виявити типові недоліки інтерфейсу та сформувати вимоги до розроблюваної системи;

– спроєктовано клієнт-серверну архітектуру вебзастосунку та визначено структуру клієнтської й серверної частин, що забезпечило логічний розподіл відповідальності між компонентами системи;

– спроєктовано структуру бази даних та взаємозв’язки між основними колекціями, що дозволило організувати централізоване зберігання даних системи;

– реалізовано систему реєстрації, авторизації та керування ролями користувачів, що забезпечило диференційований доступ покупців і продавців до функцій платформи;

– реалізовано функціонал перегляду товарів, пошуку, фільтрації та роботи з каталогом, що дозволило забезпечити зручну взаємодію покупця з асортиментом платформи;

– реалізовано систему кошика покупок та повний процес оформлення замовлення з валідацією введених даних, що дозволило автоматизувати основний бізнес-процес електронної комерції;

– реалізовано модуль управління товарами для продавців із підтримкою додавання, редагування, публікації та видалення товарів;

– забезпечено зберігання та обробку зображень товарів із використанням зовнішнього сервісу Amazon S3, що дозволило розвантажити базу даних від медіафайлів;

– реалізовано систему електронних повідомлень засобами Nodemailer для автоматичного інформування користувачів після реєстрації та оформлення замовлення;

– забезпечено взаємодію між клієнтськими та серверною частинами через REST API із використанням «fetch API» та Express;

– проведено тестування основних функціональних можливостей системи та проаналізовано отримані результати.

Для побудови серверної частини використано Node.js та Express, взаємодію між компонентами організовано через REST API, дані системи збережено у Firebase Firestore, а медіафайли у хмарному сховищі Amazon S3. Функціональна структура вебзастосунку змодельована засобами UML із побудовою діаграм варіантів використання та діяльності.

Отримані результати можуть бути використані малим та середнім бізнесом для розгортання власних онлайн-магазинів, а розроблена архітектура на основі REST API, як основа для подальшого масштабування та інтеграції додаткових модулів, зокрема платіжних систем та системи відгуків.

Результати роботи апробовано у вигляді тез доповіді на XI Міжнародній науково-практичній конференції «Globalization of Scientific Knowledge: International Cooperation and Integration of Sciences», матеріали якої опубліковано у міжнародному науковому журналі «Грааль науки» № 68 [37].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шалева, О. І. (2011). *Електронна комерція*: навч. посібник. Київ: Центр учбової літератури.

2. Laudon, K. C., & Traver, C. G. (2011). *E-commerce 2011: business, technology, society* (7th ed.). Pearson Education Limited.

3. Маєвська, А. А. (Уклад.). (2010). *Електронна комерція і право*: навч.-метод. посібник. Харків.

4. Turban, E., King, D., Lee, J., Liang, T.-P., & Turban, D. (2012). *Electronic commerce 2012: a managerial and social networks perspective* (7th ed.). Pearson Education.

5. Cherednichenko, O., Yanholenko, O., Liutenko, I., & Iakovleva, O. (2013). Monitoring and evaluation problems in higher education: Comprehensive assessment framework development. In *Proceedings of the 5th International Conference on Computer Supported Education (CSEDU 2013)* (pp. 455–460). SCITEPRESS.

6. Krug, S. (2006). *Don't make me think: a common sense approach to web usability* (2nd ed.). New Riders.

7. Lutterodt, E., & Bacconnier, L. (2024). *Personalized navigation menus on e-commerce business websites*. Jönköping University.

8. Daradkeh Y. I., and Tvoroshenko I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic, *International Journal of Advanced Computer Science and Applications*, *11*(5), pp. 43–50.

9. Змерзлий, І. Клієнт-серверна архітектура та ролі серверів. URL: https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-ролі-серверів-9893d8048229 (дата звернення: 10.05.2026).

10. WebsCraft. Взаємодія мікросервісів: request/response архітектура. URL: https://webscraft.org/blog/vzayemodiya-mikroservisiv-requestresponse-webcraft (дата звернення: 29.04.2026).

11. Mozilla Developer Network. An overview of HTTP. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (дата звернення: 18.05.2026).

12. Fowler, M. (2003). *Patterns of enterprise application architecture*. Addison-Wesley.

13. Tvoroshenko, I., & Almakaieva, A. (2020). Application of procedural generation of game content using software algorithms. In *Integration of scientific bases into practice: Abstracts of IV International Scientific and Practical Conference* (pp. 449–454). International Science Group.

14. Mozilla Developer Network. Front-end web developer. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Front-end_developer_course (дата звернення: 27.04.2026).

15. Felke-Morris, T. A. (2016). *Web development and design foundations with HTML5* (8th ed.). Pearson.

16. Ugwumba, N. (2025). *Master backend development with Node.js & Express: complete guide to REST APIs, databases & security* (1st ed., Vol. 3). University of Port Harcourt.

17. IBM. What is a REST API? URL: https://www.ibm.com/think/topics/rest-apis (дата звернення: 01.05.2026).

18. Руденко, Д. О., & Маренич, В. В. (2024). Дослідження методів розробки програмних додатків з інтегрованими API. In *Proceedings of the 2nd International Scientific and Practical Conference «Scientific Research: Modern Challenges and Future Prospects»* (pp. 144–147). MDPC Publishing.

19. Супрун, А. Є. Розробка вебзастосунку «Соціальна мережа для публікації творчих робіт»: кваліфікаційна робота першого (бакалаврського) рівня вищої освіти: спец. 122 Комп’ютерні науки. Харківський національний університет радіоелектроніки. Харків, 2024. 65 с.

20. Дацок, Є. О., Дацок, О. М., & Руденко, Д. О. (2025). Аналіз ефективності використання ORM Dapper та принципів SOLID у проєктуванні

інформаційної системи медичного призначення. *Вісник НТУ «ХПІ». Серія: Інформатика та моделювання*, *2*(14), 157–171.

21. Mozilla Developer Network. HTTP request methods. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods (дата звернення: 27.04.2026).

22. Творошенко, І. С. (2021). *Технології прийняття рішень в інформаційних системах*: навч. посібник. Харків: ХНУРЕ.

23. Ahmad M. A., Tvoroshenko I., Baker J. H., and Lyashenko V. (2019). Modeling the structure of intellectual means of decision-making using a system-oriented NFO approach, *International Journal of Emerging Trends in Engineering Research*, *7*(11), pp. 460–465.

24. Кучеренко Є. І., Творошенко І. С., Анопрієнко Т. В. (2016). Моделювання та оцінювання станів складних об'єктів із застосуванням формальної логіки, *Системи обробки інформації*, *2*(139), С. 76–82.

25. Соколова, А. М., & Руденко, Д. О. (2025). Порівняльний аналіз код-стайл гайдів різних мов програмування. In *29-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»: зб. матеріалів форуму* (Т. 7, с. 130). ХНУРЕ.

26. Tvoroshenko, I., & Andrieieva, A. (2021). Development of web applications for remote learning of English. In *Multidisciplinary academic research and innovation: Abstracts of XXVII International Scientific and Practical Conference* (pp. 645–650). International Science Group.

27. Fowler, M. (2004). *UML distilled: a brief guide to the standard object modeling language* (3rd ed.). Addison-Wesley.

28. Лисенко, Д. О., & Руденко, Д. О. (2025). Методики чистого коду та їх користь для програмістів. In *29-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»: зб. матеріалів форуму* (Т. 7, с. 80). ХНУРЕ.

29. Танянський О., Руденко Д. (2018). Порівняльний аналіз популярних JavaScript-фреймворків та бібліотек для front-end розробки. In *Інформаційні системи та технології ICT-2018* (секція 6, с. 347–349).

30. Кочкін А., Руденко Д. (2018). Використання фреймворку Angular для розробки веб-застосунків. In *Інформаційні системи та технології ICT-2018* (секція 6, с. 317–318).

31. Herron, D. (2016). *Node.js web development: create real-time server-side applications with this practical, step-by-step guide*. Packt Publishing.

32. Richardson, L., & Ruby, S. (2007). *RESTful web services*. O'Reilly and Associates.

33. Руденко Д. О., Полозова О. О. (2022). Захист інформації в контексті забезпечення економічної безпеки підприємства. *Збірник наукових праць*, с. 132–134.

34. Гороховатський В. О., Творошенко І. С. (2021). *Методи інтелектуального аналізу та оброблення даних*: навч. посібник. Харків: ХНУРЕ.

35. Гороховатський В. О., Творошенко І. С., Чмутов Ю. В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, *6*(3), С. 5–12.

36. Шафроненко А. Ю., Бодянський Є. В., Руденко Д. О., Шафроненко Є. О. (2026). Онлайн беггінг в задачах нечіткої кластеризації, *Збірник наукових праць Харківського національного університету Повітряних Сил*, *1*(87), С. 84–89.

37. Об'єздова, А. В. (2026). Порівняльний аналіз моделей електронної комерції на прикладах Amazon, OLX та Prom.ua. *Грааль науки*, *68*, 1087–1089.